PROJECT MERCURY TMY-55025

EXTERNAL SYSTEM PROGRAMS

MARCH 15, 1964



GODDARD SPACE FLIGHT CENTER
GREENBELT, MARYLAND

N 65-88098	
(ACCESSION NUMBER)	(THRU)
(PAGES)	(COBE)
(NASA CR OR TMX OR AD NUMBER)	(CATEGORY)

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

GODDARD SPACE FLIGHT CENTER GREENBELT, MARYLAND

FOR OFFICIAL USE ONLY

This document describes in detail the Goddard Mercury Real Time Programming system in effect as of August 1, 1963, and reflects the Computing program used to support the MA-9 Mercury Mission. It replaces manuals MC-107, External System Programs; MC-108, Computer Simulation Programs; and MC-110, Postflight Reporter Program.

Prepared by IBM for:

Data Operations Branch Manned Space Flight Support Division Tracking & Data Systems Directorate NASA-GODDARD

Released by:

J./J. Donegan, Head Data Operations Branch

PREFACE

This manual, EXTERNAL SYSTEM PROGRAMS, MC-63-4, is the fourth of four volumes which describe the Project Mercury Real-Time Programming System. MC-63-4 supersedes the EXTERNAL SYSTEM PROGRAMS manual, MC-107; the SIMULATION PROGRAMS manual, MC-108; and the POSTFLIGHT REPORTER PROGRAM manual, MC-110, all of which were published in August 1961 and subsequently revised in July 1962. All program descriptions presented in MC-63-4 reflect the Mercury computational system in use during the MA-9 mission of May 15, 1963.

Programs considered to be external to the Mercury operational system are those which are neither monitor programs nor processing programs. External programs complement the Mercury operational programming system and may be entered before, during, or after the mission to perform such functions as: compiling the Mercury program, simulating the real-time environment, writing the operational system tape, dumping selected portions of core, or analyzing post-mission data.

Text is arranged in five sections: SOS System for Mercury, Simulation Programs, Utility Programs, System Supporting Programs, and Postflight Analysis Programs. An index to all programs in the four manuals is included to allow cross-referencing.

TMX- 55025 MC 63-4

project **mercury**

EXTERNAL SYSTEM PROGRAMS

prepared for

National Aeronautics and Space Administration

Contract No. NAS 5-3486

revised 15 march 1964

Federal Systems Division
International Business Machines Corporation

LIST OF EFFECTIVE PAGES	
• The asterisk indica	ates pages changed, added, or deleted by the current change.

TABLE OF CONTENTS

		Page
Section 1	SOS SYSTEM FOR PROJECT MERCURY	
		1-1
1.1	Introduction to SOS	1-1
1.2	SOS Modified for Mercury	1-59
1.3	SOS Library Tape	1-81
Section 2	SIMULATION PROGRAMS	2-1
2.1	OBSERVER Program (OBSER)	2-2
2.2	HB Subroutine	2-17
2.3	HC Subroutine	2-21
2.4	RAE Subroutine	2-21
2.5	RFBRAE Subroutine	2-21
2.6	RNRCRD Subroutine	2-21
2.7	RVCAL Subroutine	2-27
2.8	SELECTOR Program	2-31
2.9	SHRED Program	2-31
2.10	SORT Program	2- 1 1 2-53
2.11	MERGE Program	2-65
2.12	Simulated Input/Output Control Program (SIC)	2-69
2.13	Conversion of the Real-Time Mercury System for Operation with SIC	2-09
2.14	Open Loop Simulation Program (OLS1)—Mercury Control Center	2-83
2.15	Closed Loop Simulation Program (CLS3)—Mercury Control Center	2-89

TABLE OF CONTENTS (Continued)

		Page
2.16	Closed Loop Simulation Program (BCLS2)—Bermuda to Goddard	2-97
2.17	Read Low-Speed SIC Tape (RLSST)	2-101
Section 3	UTILITY PROGRAMS	3-1
3.1	Program to Print Selected DCC Subchannel Input-Output Data from a Mercury Log Tape (MXCHER)	3-3
3.2	Program to Print Mercury Log Tape in Octal (MXPOCL)	3-11
3.3	Program to Print Real Time Core'd Output (MXILCO)	3-21
3.4	Symbolic Tape Updating Program (COL8ER)	3-29
3.5	Core Mapping Program (CORMAP)	3-95
3.6	Priority Indicator Listing Program (MXNDKT)	3-97
3.7	CHECKSUM Correction Program (SQZSUM)	3-99
3.8	Tape Key Comparison Program (KEYS)	3-101
3.9	Low Core Reference Program (LOWCOR)	3-103
3.10	SQUOZE Deck Comparison Program (COMPAR)	3-105
3.11	Symbolic Core Dump Program (MXCORE)	3-107
3.12	SQUOZE Tape Modification Limits Program (SUMARY)	3-111
3.13	Paper Tape Input Preparation (PAPTAP)	3-123
3.14	Low-Speed Output Printer Program (MXTHLG)	3-129
3.15	Log Tape High-Speed Input Program (HSIN7)	3-141
3.16	Log Tape High-Speed Output Program (MXHSPR)	3-147
3.17	Log Tape Printer Program (MXPRLG)	3-157
3.18	Log Tape Plotting Program (MXHSPL)	3-163
Section 4	SUPPORTING PROGRAMS	4-1
4.1	Monitor Merge Program (MXMRGE)	4-3
4.2	Mercury System Tape Writer Program (MXSTW1)	4-17
4.3	Mercury System Tape Loader (MXLOAD)	4-29
4 4	Extended Definition of Symbols Processor (MXDEFN)	4-47

TABLE OF CONTENTS (Continued)

			Page
	4.5	System Communication During Dual Compilation (SETORG)	4-55
	4.6	Real Time Transfer Trapping Test Program (MTTEST)	4-59
	4.7	Program to Write the Isolated Writer Portion of the B4 Tape (WRTB4T)	4-75
	4.8	Program to Write Dumping Portion of B4 Tape (HOMER)	4-77
	4.9	Isolated Dumping Portion of B4 Tape (ISODMP)	4-79
	4.10	Dump Program Reader (SGENDX)	4-85
	4.11	Program to Initiate Taking Snap Dumps of the Mercury System (CORING)	4-89
	4.12	Message Tape Writer Program (MXWMOT)	4-93
	4.13	Station Characteristics Tape Writer Program (U0STCH)	4-105
	4.14	Station Characteristics Tape Updating Program (U0STUP)	4-113
Sect	ion 5	POSTFLIGHT ANALYSIS AND REPORTS	5-1
	5.1	Postflight Monitor Program	5-3
	5.2	BCD Output Initialization Program (CHUMLY)	5-17
	5.3	Constant Factors Initialization Program (ACTORS)	5-17
	5.4	System Parameter Initialization Program (INITIA)	5-17
	5.5	Log Tape Sort Program (SORTER)	5-21
	5.6	Subchannel 1 Processing Program (GEB)	5-27
	5.7	IP 7094 High-Speed Input Processor Program (IPORR)	5-33
	5.8	Manual Insertion Processor Program (MANIN)	5-39
	5.9	High-Speed Output Tape Writer Program (HSOP1)	5-43
	5.10	BCD Word Conversion Program (A) (BCTB/BCTB1)	5-53
	5.11	BCD Word Conversion Program (B) (BCTB1/BCTBJ)	5-55
	5.12	Time Word Conversion Program (A) (TISWS)	5-57
	5.13	Time Word Conversion Program (B) (HMSTS)	5-59
	5.14	Unit Conversion Program (GCNVE)	5-61

TABLE OF CONTENTS (Continued)

		Page
5.15	Discrete Event Processor Program (GETME)	5-67
5.16	High-Speed Output Processor Program (DONOUT)	5-71
5.17	High-Speed Input Processor Program (DONIN)	5-81
5.18	Launch Phase Processor Program (LAUNCH)	5-85
5.19	Orbit Phase Processor Program (ORBIT)	5-93
5.20	Reentry Phase Processor Program (RENTER)	5-101
5.21	Numerical Integration Program (NUMIN)	5-107
5.22	IP 7094 Reference Frame Conversion Program (IPCNV)	5-111
5.23	B-GE Reference Frame Conversion Program (GECNV)	5-113
5.24	True Inertial Coordinate Conversion Program (MERCNV)	5-118
5.25	Atmospheric Density Processor Program (ATMOS)	5-119
5.26	Stagnation Heat Rate Processor Program (HEAT)	5-123
5.27	Range from Launch Pad Processor Program (RFLP)	5-125
5.28	Langrangian Interpolation (GRUNGY)	5-127
5.29	Recovery Area Conversion Routine (RCACNV)	5-129
5.30	Distance Computation of Earth and Space Track (DISTAN)	5-131
5.31	Utility Programs	5-137
5.32	Program Operating Procedures	5-143
Appendix A	PROGRAM INDEX	A-1
Appendix B	POSTFLIGHT REPORTER SYMBOLIC DESIGNATIONS	B-1
Appendix C	COORDINATE CONVERSION SYSTEMS	C-1
Appendix D	REPORT DATA FORMATS	D-1

ILLUSTRATIONS

Figure		Page
1-1	UA1LSC Subroutine Flow Chart	1-107
1-2	U7INTP Subroutine Flow Chart	1-113
1-3	C9RVTH Subroutine Flow Chart	1-124
1-4	LBRWR Program Flow Chart	1-131
2-1	Observer Program Flow Chart	2-10
2-2	HB Flow Chart	2-19
2-3	HC Flow Chart	2-22
2-4	RAE Flow Chart	2-23
2-5	RFBRAE Flow Chart	2-24
2-6	RNRCRD Flow Chart	2-26
2-7	Selector Program Flow Chart	2-35
2-8	Schematic Diagram (Shred Tables)	2-50
2-9	SORT Program Flow Chart	2-60
2-10	MERGE Program Flow Chart	2-67
2-11	Simulated Input/Output Control Program (SIC)	2-75
2-12	OLS1 Program Flow Chart	2-85
2-13	CLS3 Program Flow Chart	2-91
2-14	IP 7094 Data, Mercury Control Center-to-Goddard Message Format	2-93
2-15	B-GE Data, Mercury Control Center-to-Goddard Message Format	2-95
2-16	BCLS2 Program Flow Chart	2-99
2-17	High Speed Bermuda Input Format From DCC	2-102
3-1	MXCHER Program Flow Chart	3-5
3-2	MXPOCL Program Flow Chart	3-14
3-3	MXILCO Program Flow Chart	3-22
3-4	COL8ER Program Flow Chart	3-43
3-5	LOWCOR Program Flow Chart	3-104

ILLUSTRATIONS (Continued)

Figure		Page
3-6	MXCORE Program Flow Chart	3-109
3-7	SUMARY Program Flow Chart	3-121
3-8	PAPTAP. A Program Flow Chart	3-126
3-9	PAPTAP. B Program Flow Chart	3-127
3-10	MXTHLG Program Flow Chart	3-131
3-11	HSIN7 Program Flow Chart	3-143
3-12	Examples of Logging Message Formats for HSIN7	3-144
3-13	MXHSPR Program Flow Chart	3-155
3-14	MXPRLG Program Flow Chart	3-160
3-15	MXHSPL Program Flow Chart	3-165
4-1	MXMRGE General Flow Diagram	4-10
4-2	MXMRGE Program Flow Chart	4-11
4-3	MXSTW1 Program Flow Chart	4-19
4-4	MXLOAD Program Flow Chart	4-33
4-5	MXDEFN Program Flow Chart	4-51
4-6	SETORG Flow Chart	4-57
4-7	MTTEST Program Flow Chart	4-65
4-8	ISODMP Program Flow Chart	4-82
4-9	SGENDX Program Flow Chart	4-87
4-10	CORING Program Flow Chart	4-90
4-11	MXWMOT Program Flow Chart	4-96
4-12	U0STCH Program Flow Chart	4-111
4-13	U0STUT General Flow Diagram	4-118
4-14	U0STUP Program Flow Chart	4-119
5-1	Postflight Reporter Program General Flow Chart	5-10
5-2	Postflight Monitor Program Flow Chart	5-12
5- 3	CHUMLY Program Flow Chart	5-20
5-4	SORTER Program Flow Chart	5-23

ILLUSTRATIONS (Continued)

Figui	re	
		$\underline{\text{Page}}$
5-5	GEB Program Flow Chart	5-29
5 - 6	IPORR Program Flow Chart	5-34
5-7	MANIN Program Flow Chart	5-40
5-8	HOSP1 Program Flow Chart	5-45
5-9	BCTB/BCTB1 Program Flow Chart	5-54
5-10	BCTBI/BCTBJ Program Flow Chart	5-56
5–11		5 - 58
5-12	HMSTS Program Flow Chart	
5-13	GCNVE Program Flow Chart	5-60
5 - 14	GETME Flow Chart	5-62
5-15	DONOUT Program Flow Chart	5-69
5-16	DONIN Program Flow Chart	5-74
5-17	LAUNCH Program Flow Chart	5-83
5-18	ORBIT Program Flow Chart	5-88
5-19	RENTER Program Flow Chart	5-99
5-20	NUMIN Program Flow Chart	5–105
5-21	IPCNV Program Flow Chart	5–109
5-22	GECNV Program Flow Chart	5-112
5-23	MERCNV Program Flow Chart	5-114
5-24	ATMOS Program Flow Chart	5–117
5-25	HEAT Program Flow Chart	5-121
5-26	RFLP Program Flow Chart	5-124
5-27	GRUNGY Program Flow Chart	5-126
5-28	RCACNV Program Flow Chart	5-128
5-29	DISTAN Program Flow Chart	5-130
5-30	UTILITY Programs Flow Chart	5-134
5-31	DATA Deck Setup	5-139
	Door bomb	5-151

TABLES

<u>Table</u>		Page
1-1	Individual Files, Mercury SOS System Tape	1-73
1-2	IBWR1 Program Stops	1-75
1-3	IBWR2 Program Stops	1-78
1-4	Mercury SOS Library Tape Programs	1-82
2-1	Tape Format Logical Record Shred Output, High Speed Radar for Goddard	2-56
2-2	Tape Format Logical Record Shred Output, IP7094	2-57
2-3	Tape Format Logical Record Shred Output, GE-Burroughs	2-58
2-4	Tape Format Logical Record Shred Output, Low Speed TTY	2-59
2-5	Index of Routines and Subroutines Used in SIC	2-74
3-1	Telemetry Format, IP7094 and B-GE	3-145
3-2	Goddard Teletype Input	3-158
4-1	On-Line Messages	4-97
4-2	Station Characteristics Block Contents	4-107
4-3	U0STCH Station Identification	4-108
4-4	Format, Station Characteristics Changes	4-114
4-5	Card Sequence, Station Characteristics Changes	4-115

Section 1

SOS SYSTEM FOR PROJECT MERCURY

The Share Operating System (SOS) was chosen as the standard programming system for Project Mercury because of its adaptability to the varying conditions imposed on a real-time system. This section introduces SOS and contains a discussion of the differences between the SOS and Share systems and of the modifications made to SOS to meet the special characteristics of the Mercury Programming System. Also included are descriptions of the programs used to write and edit the SOS System tape (IBWR1 and IBWR2, respectively) together with program descriptions of the Mercury SOS Library tape and of utility computational subroutines.

1.1 INTRODUCTION TO SOS

1.1.1 SOS COMPILER

The Compiler in the Share Operating System has three functions: to translate, compile, and assemble. It processes the source program, written in symbolic language, and produces a tightly encoded binary deck.

Input to Compiler can be symbolic records, library routines or previously compiled programs combined with subsequent symbolic programs. The output is a squoze deck of the compiled source program. The name "squoze," adopted for the output deck, is meant to convey compactness. A squoze deck contains the source program coded in a compact form which retains the original symbolic information. It is this symbolic output that is loaded, modified, translated into actual machine language, and executed by the Modify and Load section of the SOS System.

1.1.1.1 Share Symbolic Language (SCAT)

The mnemonic term SCAT is a contraction of Share Compiler, Assembler and Translator and is widely used as the name for the symbolic language in the SOS system. SCAT is the logical extension of the Share symbolic language. The extensions which have evolved were dictated by the following general requirements:

a) The capability to recognize all machine instructions for the standard IBM 7094, for 65K core, and for all SCAT pseudo-instructions.

- b) A requirement that IBM 704 programs be compatible with SCAT. The Compiler (CP) recognizes, with some modification, the 704 symbolic language (SAP). When a SAP pseudo-instruction is different from its SCAT equivalent, the Compiler converts it to a legitimate SCAT instruction. However, SCAT/SAP compatibility does not extend beyond the compiling phase. Modify and Load accepts only legitimate SCAT codes, treating all others as illegal.
- c) The incorporation of variable-length mnemonics, which facilitates consistent channel designation and provides a convenient means of specifying macro-instructions to be processed by the Compiler.
- d) For ease of key punching, the variable field should begin in the same column of every card, regardless of the length of the mnemonic.

1.1.1.2 Symbolic Input Format

The format of the symbolic instruction, with fields fixed at their maximum limits, is:

Card Columns	Description
1 - 6	Location field or blank
7	Blank
8 - 14	Operation code (including asterisk for indirect addressing)
15	Blank
16 - 72	Variable field and comments, which must be separated by a blank
73 - 80	Not Used

Therefore, the mnemonic operation code (beginning in card column 8) may be from one to six letters in length. At least one blank must follow the last letter; the number of blanks that may follow must be such that the length of the operation code plus the number of blanks is less than or equal to eight. If the variable field does not begin by column 16, it is assumed to be blank.

Four principal parts of a symbolic instruction are recognized: location symbol, operation code, variable field, and comment field. The location symbol is a name for either a storage location or other expression associated with the instruction; the precise item named is dependent upon the operation specified. In all cases, the operation determines the nature of the instruction and

guides the interpretation of the various parts. The variable field is construed in many ways as a function of the operation part of the instruction. In general, with the location symbol and operation, the variable field gives the complete instruction specifications. The comment field has the sole function of describing a remark intended to appear on a listing and is not pertinent to the running of the program.

The order for the variable field of a 709/7090/7094 symbolic instruction is address, tag, decrement. These subfields within the variable field are separated by commas. In all instructions it is possible to omit parts of the variable field. To omit only an interior part (the tag, for example) it is necessary to have two commas in adjacent positions, because the first blank encountered in a variable field terminates that field. TXI A, 0, B and TXI A,, B result in the same word. Comments may begin after a blank indicating the end of the variable field; however, for ease in key punching and to maintain uniformity, comments should begin in column 35. Comments may not begin before column 17.

1.1.1.3 Symbolic Language and Arithmetic Expressions

The basic units of the symbolic language are symbols, numbers, and operation codes. These units may be combined by punctuation marks, subject to certain rules, to yield expressions.

A symbol is a combination of from one to six Hollerith characters, at least one of which is nonnumeric and none are a +, -, *, ?, \$, =, comma, or an imbedded blank. A blank is not considered a character in this case. A symbol is defined only if it appears in the location field of some instruction; otherwise, it is undefined. A symbolic instruction should have a location symbol only when it is necessary to refer to that instruction in the program. An absolute location symbol, i.e., one containing only numeric characters, is flagged as an error and is ignored. Leading zeroes are considered legitimate characters of a symbol.

A number is a combination of digits which may be decimal or octal, depending upon the operation code of the instruction in which it appears. An operation code may consist of from three to six alphabetic characters. An expression is a combination of symbols and integers separated by the following connectors or punctuation marks:

+ addition

* multiplication

- subtraction

/ division

These connectors have different meanings when used in the BOOL pseudooperation (which is defined later).

1.1.1.4 Evaluation of Variable Field Expressions

Constants in a variable field must be less than 2^{35} . They are considered decimal quantities unless the instruction is a Type D instruction. Examples of Type D instructions are: RIL 1, RIR 44, SIL 1 and LFT 2. The constants of a Type D instruction are treated as octal values. Only simple expressions are permissible in the variable field of these instructions, and the value is computed modulo 2^{18} . With all other instruction types, if the symbol referred to in a simple expression is octal (Boolean), the address and decrement fields are treated as 18-bit values and the tag is computed modulo 8. When not octal, the address and decrement fields are considered as 15-bit values and the tag is computed modulo 8.

1.1.1.5 Special Characters

The asterisk (*) character has five different meanings in SCAT depending upon context. As a punch in column 1 of the card, the asterisk defines the card as a remark or comment card. If it is found immediately after an operation code, it specifies indirect addressing. As a connector in a variable field expression, it connotes multiplication. As a Boolean operator, it specifies intersection, e.g., the logical AND process. Finally, if it occurs immediately after another connector or as the first character in a variable field, it must be recognized as a term. In this context an asterisk is interpreted as having the current value of the location counter.

The dollar sign character (\$) may be preceded by a numerical, alphabetic, or special character, or it may start a term followed by five or fewer characters in an expression. These collocations cause SCAT to head the symbol with the given character rather than the current heading character. Reference from a headed region to an unheaded symbol is made by preceding the \$ with no heading character. Previously, such referencing was also possible by preceding a \$ with zero.

1.1.1.6 Classification of Operation Codes

There are two classifications of instructions: machine and nonmachine. The latter type are collectively called pseudo-instructions. For purposes of this discussion, the pseudo-instructions are arbitrarily divided into three categories, one of which retains the generic name pseudo-instruction. The Compiler, therefore, recognizes four classes of instructions:

- a) Machine instructions
- b) Pseudo-instructions
- c) Macro-instructions
- d) List control pseudo-instructions

1.1.1.7 Machine Instructions

A machine instruction (i.e., an instruction using a machine operation) always generates one 36-bit binary machine word in the object program. The rules for specifying the location field and the variable field of a machine instruction have been stated previously. The vocabulary of 709/7094 instructions and their SCAT mnemonics appears in subsection 1.1.1.11. (For information concerning the operation of these instructions, refer to the 709/7094 Reference Manual.)

1.1.1.8 Pseudo-Instructions

Unlike machine instructions, some pseudo-instructions may generate more than one machine word in the object program; others generate no words at all. The pseudo-operations of SOS have a variety of functions.

The remainder of this section describes the pseudo-operations of the Compiler section of SOS (except for those which direct the Modify and Load program).

The mnemonics L and VF in the following paragraphs refer to location counter and variable field, respectively.

Assignment of Absolute Storage Locations-Origin (ORG)

The basic function of an assembly process is to assign absolute storage locations to machine instructions. However, there must be an address at which this assignment begins. In SCAT this value is furnished to the assembly program by the program being assembled via the ORG pseudo-instruction. ORG sets the location counter to the same integer value as that computed for its variable field. A location symbol associated with an ORG instruction is also assigned the computed value of the variable field:

		Address,
		Tag,
Location	Operation	Decrement
	ORG	100

In the example above, ORG assigns a value of 100_{10} to the location counter. The location counter determines the storage location to which the subsequent instructions are assigned. The first instruction following the ORG card is assigned the location of the variable field value, modulo 2^{15} , of the ORG card.

A symbol appearing in the variable field expression need not have been defined previously, i.e., it need not have appeared in the location field (columns

1-6) of some previous instruction or pseudo-instruction. However, a symbol in the expression which is not eventually defined in the program renders the variable field of ORG noncomputable.

If the program being assembled does not have an ORG pseudo-instruction, Modify and Load sets the ORG to the lowest location in memory not required by the SCAT system (3000₁₀). Subroutines assembled without ORG can be inserted into a job where needed as long as they are prefaced by a SQZ control card.

Floating Origins—SYSFLO Table and QORGN Macro

When programs share common storage with either a permanent replacement (as in real-time loading) or a temporary replacement (as in real-time buffering) more efficient allocation of core storage may be accomplished if the numeric equivalents of symbolic locations in job 1 can be made available to the SOS system during the writing of the job 2, B1 (actually B3) tape. Local modifications have been made to the SOS system tapes used for Project Mercury which enable information to be transferred from one job to a successive job via a table within the IBMonitor section of SOS. Since this table must be maintained and used in IBMonitor during the Compile or Modify and Load run from job 1 through the last job, any reinitialization of IBMonitor between Mercury dual-compilation jobs would destroy this table.

a) SYSFLO Table—a 14-cell table in IBMonitor, beginning with and referred to by the symbolic location SYSFLO, provides storage for twenty-eight 15-bit values. Each word may contain two values—one in the decrement, the other in the address, etc., from SYSFLO through SYSFLO + 13.

When SOS is compiling or modifying a job for a Load and Go run, the first TCD (transfer card) pseudo-operation, followed immediately by a BSS or BES pseudo-operation, causes SOS to equate the symbol associated with the BSS or BES to the value contained in the address field of the location SYSFLO. Successive combinations of TCD and BSS or BES pseudo-ops force SOS to extract successive values from the table. As each value is used, the field containing that value is cleared. The priority of SYSFLO is given by:

	Decrement	Address
SYSFLO	2	1
+ 1	4	3
+ 2	6	5
•	•	•
•	•	•
•	•	•
+ 13	28	27

Since the first zero field signals the effective end of the SYSFLO values, the entries must be packed from the top and a field of zero may not be transferred between jobs via SYSFLO. In addition, between any two jobs, the values in SYSFLO must be set in correspondence to the SOS processing order of the TCD-BSS (or BES) combinations in the successive job which is to produce the SYSFLO reference.

After the final value is used, SOS flags the first location in the table to indicate that the table has been exhausted. All successive TCD and BSS combinations are now handled by SOS to origin the symbol at the value contained in SOS's SYSORG. SYSORG contains in the address the value of the first available location after the SOS reserved area of core.

When the value from job 1 has been associated with a suitable symbol in job 2, this value may be used for any purpose. An example of SYSFLO usage may be illustrated by the QORGN macro. The SYSFLO value is used by QORGN in determining an origin which floats, such that a program in job 2 may be origined after the longer of two programs sharing the same core area with one of the programs being in job 1.

b) QORGN Macro—QORGN determines the longer of two (and may be expanded to determine the longer of three or more) programs and provides a symbolic location which may be used to origin a third program after the longer of the previous two programs. QORGN is used to provide automatic origins for the Mercury Programming System's real-time loading and buffering.

The QORGN macro is defined:

QORGN	MACRO	A, B, BP1, AF, BF, ORGSYM
BP1	EQU	B + 1
\mathbf{AF}	EQU	B/A + 32767
\mathbf{BF}	EQU	A/BP1 + 32767
	ORG	AF/32767*A + BF/32767*B
ORGSYM	PZE	0
	TCD	ORGSYM
Α	BSS	0
	HTR	0
	END	

Where symbol B is the last symbol associated with a particular program, or package, i.e., LCSYMB, in job 2, and symbol A are the symbols to which SOS gives a value from the SYSFLO table in IBMonitor. ORGSYM is set equal to the larger of A and B and is therefore usable

as a symbolic origin point. ORGSYM receives a value equal to the larger of A or B. BP1, AF, and BF are symbols internal to the macro. These symbols must be uniquely defined in the compilation.

QORGN is executed only by SOS and may be used only by an SOS system especially modified for this purpose. QORGN is not executed by the operational tracking program. The files generated by QORGN are used only to provide interjob communication to SOS and have no function in the absolute Mercury program tapes.

Block Started by Symbol (BSS)

A BSS can occur anywhere in a program. This pseudo-instruction reserves a block of storage when the program being assembled demands it. The block reserved is equal in length to the value of the variable field expression and extends from L to L+(VF-1). The associated location symbol is given the value that L has when it encounters the BSS and corresponds, therefore, to the first word of the block reserved.

Location Counter	Location	Operation	Address, Tag, Decrement
250	Α	BSS	200
4 50	В	XXX	XXX

In the example above, the BSS instruction reserves the 200 memory positions from locations 250 to 449, inclusive. The location symbol A is assigned to the value 250.

The rules for previous definition of symbols are the same as for the ORG pseudo-instruction.

Block Ended by Symbol (BES)

A BES may occur anywhere in a program. This pseudo-instruction reserves a block of storage at the direction of the program being assembled. A BES is the same as a BSS in every respect except for its result on the associated location symbol. This symbol is given the value L + VF and corresponds to the first word following the block reserved. Whereas the associated location symbol in a BSS has the value of L, it is assigned the value of L + VF in a BES instruction.

The rules for previous definition of symbols are the same as for the ORG pseudo-instruction.

The variable field of a BSS or BES may specify, as a tag, a code indicating the format of the data to be stored ultimately in the reserved block of storage. This specification is not required, but enables debugging programs to make a meaningful listing of such data. The codes are:

F-Floating-point numbers

X-Fixed-point numbers

O—Octal data

H-Hollerith (binary coded decimal data)

S-Symbolic instruction

C-I/O command

V-Variable Field Definition (VFD)

For example, a programmer writes:

Location	Operation	Variable Field
ALPHA	BSS	50, F

By using F, he is saying to the debugging system: "The 50 cells in the block beginning at ALPHA are to be interpreted as containing floating-point numbers when I ask you later to give me the contents of any of these cells."

Transfer Card (TCD)

This pseudo-instruction directs the loading program to transfer control to the program being loaded. The transfer is made to the storage location represented by the value of the variable field expression of the TCD instruction.

There can be more than one TCD instruction which may appear anywhere in the program.

If a TCD has an associated location symbol, the symbol is assigned the value that L has when it encounters the TCD instruction.

Location			Address, Tag,
Counter	Location	Operation	Decrement
200	Α	TCD	2500

The instruction above sets A equal to 200; transfer of control is made to location 2500.

End (END)

Since the computer must know where to start assigning absolute storage locations to machine instructions, it must also know when to stop this process. In SCAT, the termination of the assembly and loading operations is indicated by the END pseudo-instruction. It must appear as the last instruction read during the assembly process of every program.

As is the case with a TCD, the END instruction causes a transfer of control to be made to the storage location represented by the value of the variable field expression. The rules governing the associated location symbol are the same as TCD.

			Address,
Location			Tag,
Counter	Location	Operation	Decrement
800	A	END	1000

The instruction above sets A equal to 800; transfer of control is made to location 1000.

Equal (EQU)

EQU assigns the integer value given by the expression appearing in the variable field to the symbol appearing in columns 1-6.

This pseudo-operation is used when the symbol appearing in columns 1-6 specifies a preset program parameter such as the number of items in a group or any other quantity which is invariant with respect to the location of the program in storage. If the symbol in columns 1-6 specifies the location of a piece of data or an instruction, the pseudo-instruction SYN should be used.

Synonym (SYN)

SYN assigns the integer value given by the expression appearing in the variable field to the symbol appearing in columns 1-6.

The pseudo-operation SYN is used when the symbol appearing in columns 1-6 specifies the location of a piece of data or other quantities whose values depend upon the location of the program in storage.

In SCAT language, EQU and SYN may be used interchangeably since the distinction is taken care of automatically by Modify and Load. However, EQU and SYN have different effects if the binary object program is to be produced in a relocatable binary form. For the sake of clarity and use in later compilations, the distinction between EQU and SYN should be made.

Boolean (BOOL)

The BOOL pseudo-instruction is similar to EQU and SYN in that it defines a location symbol by equating it to the value of the single expression appearing in the variable field. All numbers in the variable field must be octal. The appearance of an 8 or 9 in the variable field indicates an error, and the computed value of the field is erroneous.

Computing the value of a Boolean variable field differs from computing the value of an ordinary expression because the Boolean punctuation marks specify logical rather than arithmetic operations, and the result is expressed modulo 218.

The punctuation marks, or operators, which may be used in this pseudo-instruction are:

<u>OPERATOR</u>	<u> 1</u>	MEANING	
	Algebra of Cl	asses	709/7094
+	Union		Inclusive OR
-	Symmetric diffe	erence	Exclusive OR
*	Intersection		AND
/	Complementation	on	Complementation
For example: SYMBI	L BOOL	505*617	results in an octal 105.

As with the EQU and SYN pseudo-instructions, the BOOL instruction must have a location symbol associated with it. The variable field of this instruction must be a single expression. Any division of the field into address, tag, or decrement causes the tag or decrement parts to be ignored and results in an error indication.

If the programmer is using the sense indicator register in his source program, he may often need to write Type D instructions, the 18-bit address part of which corresponds to the 18 leftmost or rightmost bits of this special register (see pp. 51 and 60 of the 709 Reference Manual, A22 - 6501 - 1). If

the particular sense indicator positions cannot be conveniently predetermined, the instruction can be reserved by using:

Location	Operation	Variable Field
	:	:
	RIR	SENSX
	•	•

Later, when the programmer has decided that SENSX should be, for example, the rightmost four positions (i.e., positions 32, 33, 34 and 35 of the sense indicator register), he can write:

Location	Operation	Variable Field
:	:	<u>:</u>
SENSX	BOOL	17
•	:	:

The 17 is interpreted as an octal number equivalent to (000 000 000 000 001 111)2.

Heading (HEAD)

The HEAD pseudo-instruction renames symbols of fewer than six characters by inserting an additional character at the beginning of each symbol.

The variable field of a HEAD instruction must consist of only one character or a blank. Any other configuration results in an error indication and is ignored by the loading and assembly process.

The HEAD pseudo-instruction prefixes the heading character, or blank, to every location symbol and every variable field symbol of five or fewer characters encountered subsequent to itself and prior to the occurrence of another such instruction.

Location symbols and variable field symbols of six characters are not affected by the HEAD pseudo-instruction. This is significant since it is through the use of 6-character symbols and the punctuation mark, \$, that reference from one headed field to another is possible.

A dollar sign appearing in a variable field is significant for the following reasons:

a) An expression consisting of a single character followed by a \$ and a symbol of fewer than six characters is equivalent to the symbol headed by the initial character. For example, X\$A is equivalent to A headed by X. Such an expression is not affected by the HEAD pseudo-instruction.

b) An expression consisting of a \$ followed by a symbol of fewer than six characters is equivalent to the symbol headed by a blank. Such an expression is not affected by the HEAD pseudo-instruction.

The following code illustrates the considerations mentioned above:

Absolute Location	Symbolic Location	Code	Absolute Address
		<u></u>	
0	Α	CLA B	1
1	В	CLA A\$A	2
		HEAD A	
2	Α	CLA B	3
3	В	CLA \$A	0
4		CLA B\$A	5
		HEAD B	
5	Α	CLA A\$B	3
6		CLA \$X	7
		HEAD	
7		CLA A	0
8		CLA B\$A	5
9		CLA COMMON	11
		HEAD C	
10		CLA COMMON	11
11	COMMON	BSS, 1, F	11

Additional information:

- a) If no heading character is given, Compiler heads with a blank. Heading can be discontinued by using HEAD with a blank variable field.
- b) Zero is a distinct heading character and indicates a heading.
- c) Reference to a headed symbol of five characters cannot be made by compounding a 6-character symbol of the symbol and the heading character. Thus, a reference in a variable field of ABCDE headed by X must be of the form X\$ABCDE and not XABCDE.

Decimal (DEC)

This pseudo-instruction provides decimal data to the program being assembled. A single DEC instruction may specify more than one decimal number per card. Successive words are specified in the variable field and are separated by commas. The first blank encountered in the variable field terminates it. The data words generated by this instruction are assigned successively increasing

storage locations; the location symbol, if present, is assigned the value of the storage location of the first word.

The sign of a number is indicated by a plus or a minus preceding the number, exponent, or binary scale factor. The absence of any punctuation implies a plus sign.

The variable field expression of a DEC instruction must be a numerical expression. The only characters admissible in such fields are commas, numerical constants, plus (+), minus (-), period (.), E, or B.

Data generated by this pseudo-instruction is converted to one of three specified forms (binary integer, floating-point binary number, or fixed-point binary number) according to the following rules:

- a) Binary integer (with the binary point at the right end of the word) if neither a period (.), B, nor E appears in the numerical expression.
- b) Floating-point binary number if a period (.) or E, or both, but not B appears in the numerical expression. The appearance of E may be explicit or implicit.
 - 1) The decimal exponent to be used in the conversion is the number which immediately follows E. If E is not present, it may be implied by a signed number.
 - 2) The exponent is assumed to be zero if neither E nor a signed number appears.
 - 3) If the decimal point does not appear, it is assumed to be at the right end of the word.

The expressions +12.345, 12.345, 1.2345E1, 1.2345 +1, 1.2345E +1, 1234.5E-2, 1234.5-2 and 12345E-3 are all equivalent representations of the same floating-point number which is normalized following conversion.

- c) Fixed-point binary number if the character B appears in the numerical expression:
 - 1) The binary scale factor used in the conversion is the number immediately following B and may be positive, negative, or zero. (This factor is the count of binary positions between the left end and the binary point of the fixed-point binary result.)
 - 2) If the decimal point does not appear, it is assumed to be at the right end of the word.

3) The decimal exponent used in the conversion is the number immediately following E or, in the absence of E, implied by a signed number. If both B and E appear, the order of their appearance is irrelevant. For example, 1.2E1B4, 1.2B4E1, 1.2+1B4 and 1.2B4+1 are equivalent expressions.

Any word generated by a DEC pseudo-instruction which exceeds the limit of a machine cell results in a zero and an error is indicated.

In a DEC pseudo-instruction, a blank variable field, successive commas in the variable field, or a variable field ending in a comma all imply the generation of a zero.

Octal (OCT)

This pseudo-instruction provides octal data to the program being assembled. A single OCT instruction may specify more than one octal number per card. Successive words are specified in the variable field and are separated by commas. The first blank encountered in the variable field terminates it. Data words generated by this instruction are assigned successively increasing storage locations and the location symbol, if present, is assigned the value of the storage location of the first word. (OCT is similar to the DEC pseudo-instruction except for the kind of data generated.)

Octal numbers may be preceded by plus or minus signs; the absence of any sign implies a plus sign.

In an OCT pseudo-instruction, blank variable field, successive commas in the variable field, and a variable field ending in a comma all imply the generation of the zero.

Binary Coded Information (BCI)

This pseudo-instruction provides Hollerith data in standard binary coded decimal form to the program being assembled. The variable field of this instruction consists of one digit from 1-9, followed by a comma, followed by any characters (including the blank) which are acceptable to SCAT. Specified characters

following the comma are packed together six to a 709/7094 word, and these words are assigned successively increasing storage locations. The number of words generated is specified by the digit preceding the comma. If a comma does not follow the first digit of the variable field, an error indication is given. Any location symbol associated with a BCI instruction is assigned the value of the storage location of the first word generated by the instruction. The use of another BCI card is required for more than nine words.

Library (LBR)

The LBR pseudo-instruction is used to extract a subroutine from a library tape and incorporate it into the program being assembled. The complete format is:

		Address,
		Tag,
Location	Operation	Decrement
		IDENT, U, CHANNEL
SUBR	LBR	AND TAPE NUMBER

If present, the location symbol is assigned to the first instruction in the library program, provided that the first instruction is not EQU, SYN or BOOL. If the first instruction already has a location symbol, it is replaced by the location symbol of the LBR instruction.

IDENT and the Channel and Tape Number are used only to locate a subroutine in the tape library. The IDENT may be a symbol or an integer. If it is a zero or a blank, the location symbol is used as the identification. If the Channel and Tape Number are zero or blank, it is assumed that the subroutine is on the SCAT library tape, and the location symbol is used as the label in this case.

The symbol U (unrelativized) indicates to SCAT that the library subroutine is not to be relativized. If the tag field contains any other symbol or is blank, the program is to be relativized. Relativization is the process by which all addresses in the library subroutine are expressed relative to the entry points in the library program.

The SAP pseudo-instruction LIB is changed by SCAT to an LBR and executed accordingly. However, the following conditions are assumed:

- a) The subroutine is on the system tape.
- b) The subroutine is relativized.
- c) The location symbol of LIB serves as the identification of the subroutine being called for by LBR. All addresses within a subroutine are expressed relative to the base point address.

Variable Field Definition (VFD)

VFD is used to specify the division of words in other than standard prefix, decrement, tag, and address fields. The variable field consists of defining expressions, or subfields, which may specify three types of information: symbolic, octal, and/or Hollerith. These subfields within the variable field are of the following form:

$$n_1/E_1$$
, On_2/E_2 , Hn_3/E_3

In the example above, \underline{n} is a decimal constant indicating the number of bits to be occupied by the subfield; E is an ordinary variable field expression; H indicates a Hollerith subfield; and O indicates an octal subfield. All subfields are terminated by a comma or blank; these may not be included among the specified characters. If the given expression is longer than the designated \underline{n} bits, the value of the subfield is taken modulo 2^n , i.e., the rightmost \underline{n} bits are used. If it is shorter, the leftmost bits are filled in with blank characters in the case of a Hollerith subfield and with zeros for all other types of subfields.

The first subfield specified begins at the leftmost part of the first word generated. If a location symbol appears, it is equated to the location of this word. The next subfield begins to the right of the previously defined subfield. If a subfield extends beyond the end of a word, it is continued from the left end of the next word.

There is no limit to the number of subfields which may be specified by this pseudo-instruction; however, the length of any subfield cannot exceed 63 bits.

All subfields give the actual expression and not the location of the expression. All expressions are computed modulo the length of the subfield rather than in the usual manner.

The expressions of the VFD variable field may be either ordinary or Boolean, or both, but they cannot both be in the same subfield.

Unless prefixed by O, the numbers in the variable field expression are to the base 10 even when they occur with Boolean symbols.

Et Cetera (ETC)

ETC is used only to extend the variable field of the previous instruction. The variable field of the previous instruction must be terminated by a comma. If the comma has significance within the field, the break must be made at an insignificant comma. If the previous variable field does not terminate with a comma, a comma is assumed and an error is indicated. In any event, the

variable field of an ETC pseudo-instruction is considered an extension of the variable field of the previous instruction, commencing at a comma, i.e., with a complete expression.

An ETC pseudo-instruction may not have a location symbol associated with it. The following points about ETC should be clearly understood:

- a) If a comma has significance within a field which is being extended by an ETC instruction, the break must occur at a comma which separates fields, i.e., the comma signalling the ETC must not be introduced within an expression.
- b) The variable field of ETC does not begin with a comma. In fact, it does not differ from any other variable field. In the preliminary description of SCAT, it is stated that "the variable field of an ETC pseudo-instruction is considered an extension of the variable field of the previous instruction, commencing at a comma, i.e., with a complete expression." This is true but could be misleading. The critical word is "at"-the expression commences immediately after a comma, but does not include the comma.
- c) An ETC may follow only a VFD pseudo-instruction, the MACRO pseudo-instruction, or any operation code calling for the generation of a system or programmer macro, and nothing else.

Remarks (*)

The pseudo-instruction asterisk, *, (indicated in column 1 of a card) enters commentary material, which is to appear on a listing into the program being assembled. The remaining 71 positions of the symbolic card may be used as a comment field.

This pseudo-instruction has no location field (the asterisk is not recognized as such), operation code field, or variable field. It has no effect upon the assembly process.

1.1.1.9 Macro-Instructions

A macro-instruction generates a word or a sequence of words. Parameters required by the macro subroutine must appear in the variable field of the macro-instruction. These parameters are incorporated into the word or sequence of words generated by the macro-instruction during the compilation (rather than at the execution time) of the object program.

There are two types of macro-instructions in SCAT: system and programmer. System macros are provided for in the Compiler. Programmer macros are innovated in the source program.

System Macro-Instructions

The generation of a system macro-instruction is called for when its code name appears in the operation code field. The variable field specifies the parameters to be used in the generated sequence of words. Any location symbol associated with the macro-operation is assigned as the location symbol of the first of the generated words.

At present there are two macro-instructions which have been incorporated into the Compiler: BEGIN and RETURN. It is assumed that many such macro-instructions will be available in the Compiler and that others will be added by installations to perform special jobs.

- a) BEGIN K, T, I, E—the BEGIN macro-instruction generates the basic subroutine linkage recommended by the Share Operating System Committee. The parameters K, T, I and E are defined as follows:
 - K-location of the normal return relative to the TSX. The exit transfer is TRA K, 4.
 - T—specification of the index registers to be saved. Index register 4 should always be saved as a debugging aid.
 - I—If I is 1, the sense indicators are to be saved and restored; if I = 0, or blank, they are not to be saved or restored.
 - E—specifies whether to save and restore a cell to indicate what channel traps should be enabled.

The number of resulting instructions equals 2X + 3I + 2, where X is the number of index registers specified by T and I is as defined above.

The maximum and minimum sequences for the corresponding macroinstructions are given below.

Maximum Sequence:

SR	BEGIN	2, 7, 1	\mathbf{SR}	\mathtt{TXL}	*+7
		•		\mathbf{AXT}	0,4
				\mathbf{AXT}	0,2
				\mathbf{AXT}	0,1
				\mathbf{LDI}	*+2
				TRA	2.4

				PZE STI SXA SXA SXA	*-1 *-5,1 *-7,2 *-9,4
Minimun	n Sequence:				
SR	BEGIN	2,4	SR	TXL AXT TRA SXA	*+3 0,4 2,4 *-2,4

b) A RETURN SR, n—this macro-instruction specifies the error code and generates the instructions necessary for the normal and error exits from the routine. If present, A is the location of the first generated instruction; SR identifies the subroutine. This identification is necessary since RETURN need not refer to the most recent BEGIN macro-instruction. The error code, n, is stored in the decrement of the first generated instruction of the associated BEGIN.

If no error return procedure is desired, n is zero or blank. In this case, one instruction results:

TRA SR + 1

If n is specified, the following sequence is generated:

AXT n, 4 SXD SR, 4 LXA SR + 1, 4 TXI SR + 2, 4, 1

The use of the system macro-instructions is illustrated below:

	Source	Program	Object	Program
SR	BEGIN TPL	2, 7, 1 SR2	SR TXL AXT	*+7 0, 4
SR1 SR2	RETURN DVP STQ	SR, 1 X Y	AXT AXT LDI	0, 2 0, 1 *+2
SR3	RETURN	SR	TRA PZE	2, 4
			STI SXA SXA	* -1 * -5, 1 * -7, 2

Source	Program	Object	Program
		SXA	*-9, 4
		TPL SR1 AXT	SR2
		SR1 AXT SXD	1, 4 SR, 4
		LXA	SR+1, 4
		TXI	SR + 2, 4, 1
		SR2 DVP	X
		STQ	Y
		SR3 TRA	SR+1

Programmer Macro Instructions

In addition to system macro-instructions, the Compiler processes macro-instructions defined by the programmer for use in the program being compiled. The definition is introduced to the Compiler by the MACRO pseudo-instruction which must have the code name of the programmer macro in its location symbol field and the code MACRO in its operation field. The location symbol must be from one to five characters in length, must be completely alphabetic, and must not be the code name of a machine operation, a pseudo-operation, or a system macro-operation. If the given code name is that of a previously defined programmer macro-instruction, the new definition replaces the former one.

The MACRO card lists in its variable field the parameters to appear in the defining example. All these parameters must be nonconstant. The variable field may be extended by ETC cards. However, the maximum number of parameters which can be specified by a MACRO pseudo-instruction and its associated ETC cards is 32. They are separated by commas.

The instructions which constitute the defining example follow the MACRO card in a sequence terminated by an END card. A defining instruction may have in its variable field any valid combination of symbols and connectors. All location symbols are variable field symbols of the defining example and must have appeared in the parameter list of the MACRO card.

Although the example used to illustrate the technique of writing macro-instructions shows all the variable field symbols as appearing in the parameter list, it is not necessary that such symbols be included among the parameters. However, all location symbols must appear elsewhere in the program.

If symbols appear in the parameter list and elsewhere in the program, preference is given to their definitions in the parameter list in attempting to define a programmer macro-instruction.

The following example illustrates a MACRO pseudo-instruction and its definition.

POLY	MACRO	COEFF, INVAR, DPVAR, DEG
	ETC	T, Z
	\mathbf{AXT}	DEG, T
	LDQ	COEFF
DPVAR	\mathbf{FMP}	A\$INVAR
	${f Z}$	COEFF + DEG + 1, T
	XCA	
	TIX	DPVAR, T, 1
	END	

The location symbol of the MACRO pseudo-instruction becomes the operation code of the defined programmer macro-instruction. The number of instructions generated by a programmer macro-instruction is always the same as the number in the defining example. For example, the symbol POLY (defined above) could be used to form the macro-instruction:

POLY
$$C1 + 10$$
, X, FX, 3, 4, FAD

which would then generate the following sequences, or skeleton, in accordance with the pattern of the defining example:

In the coding example, the first two instructions of the defining example are:

The entire example is correct as shown. It is desirable, however, to be very explicit about the following:

A parameter used in the defining example must not be the mnemonic for any instruction.

As the example shows, it is permissible to have one of the parameters represent an operation code in the manner in which Z stands for FAD. This means that an operation code may be included among the parameters of a defined macro-instruction, as the following example illustrates:

POLY
$$C1 + 10$$
, X, FX, 3, 4, FAD

The restriction mentioned here applies only to the parameter list of the defining example.

A system macro can occur in the definition of a programmer macro; a programmer macro cannot occur in the definition of a programmer macro.

Note that the parameters of the defined macro-instruction may be symbolic or absolute, that they have a one-to-one correspondence with the dummy parameters of the MACRO pseudo-instruction, and that they have replaced the dummy parameters in the generated skeleton. Symbols which are to appear in the variable fields of the generated instructions may appear elsewhere in the source program. However, symbols to appear in the location fields of the generated instructions must not appear elsewhere in the program. This would result in multiple definition of the symbols.

Properties of Both System and Programmer Macro-Instructions

- a) A location symbol is identified with the first instruction generated.
- b) The variable field may consist of expressions and simple symbols. Any expression which ultimately appears as a divisor of a fraction in a variable field may have only one symbol.
- c) The variable field may be extended by ETC cards.

1.1.1.10 List Control Pseudo-Instructions

The Compiler provides the following as a listing: symbolic program with comments and alter and relative numbers; page heading, page number, and date on each page; an optional octal or decimal absolute program; error tables containing duplicated symbols, undefined symbols, and the total number of error-flagged instructions; and an optional symbol table which gives the symbol and page number. The list may be used in finding symbols in the listing when no absolute program is printed.

The following list control pseudo-instructions are provided to edit the listing of any program:

- a) LIST—the LIST pseudo-instruction causes printing in the normal mode—all cards are listed without printing in detail, i.e., without printing words generated by pseudo-instructions (OCT, VFD, DEC, LBR and BCI) or by macro-instructions.
- b) UNLIST—an UNLIST instruction completely suspends printing until a LIST instruction is encountered.

- c) DETAIL—if the instruction DETAIL (with a blank variable field) is encountered, any printing which is currently in progress continues with complete detail, i.e., the machine words generated by macro-instructions (system and programmer macros), LBR, DEC, OCT, BCI and VFD instructions, are printed. The effect of a DETAIL instruction is nullified when a TITLE, LIST, or UNLIST instruction is encountered.
- d) TITLE—a TITLE instruction causes any printing currently in progress to be continued in the normal mode (i.e., without any detail) until a subsequent DETAIL instruction or an UNLIST instruction is encountered. If printing is already in progress in the normal mode, or if no printing is in progress at all, a TITLE instruction has no effect.

1.1.1.11 SCAT 709/7094 Machine Instructions

Included in the list of machine instructions (although they are not actually machine instructions) are those operation codes which may be used to assign arbitrary values to the prefix and sign of calling sequence words. They are listed as a group below:

Alphabetic Code	Name	Octal Code
MZE	Minus zero	-0
MON	Minus one	-1
MTW	Minus two	-2
MTH	Minus three	-3
PZE	Plus zero	+0
PON	Plus one	+1
PTW	Plus two	+2
PTH	Plus three	+3
FOR	Four	-0
FVE	Five	-1
SIX	Six	-2
SVN	Seven	-3

The codes listed below with information concerning address (A), tag (T), decrement (D), and indirect addressing (I) appear under various headings and are defined as follows:

N—this entry under the columns A, T, D and I indicates that the corresponding instruction should not have an address, tag, decrement or indirect address, respectively. A zero in the address, tag, or decrement does not violate this restriction. If the prescribed field is specified, it is processed as given and an error is noted.

- Y—this entry under a column heading indicates that the specified parts of the corresponding instruction should occur. If the field is to be provided by the program, a zero should be used.
- O-this heading under column A indicates that the address field must be an octal number or Boolean symbol.
- I—this entry under column T indicates that the tag field, if specified, must be a 1 or an expression with an equivalence of 1. No other non-zero tag is permitted.
- C-there are six instructions (CAQ, CRQ, CVR, VDH, VDP, VLM) which use the decrement field as a count. C appears under column D of these instructions to indicate that the count is required.

1.1.2 SOS PRODUCED PROGRAM LISTINGS

This subsection describes the form of program listings produced by the SOS system. The material is included here to introduce the Modify and Load pseudo-operation presented in subsection 1.1.3, since references to information in the program listings are necessary in that subsection.

The purpose of the SOS listing facilities is to provide means of obtaining necessary information when making program modifications. Listings produced by SOS are made in symbolic form since this is the most useful method for determining necessary changes.

Symbolic listings of a squoze deck reproduce, with some exceptions, the symbolic source deck program including modifications incorporated by the punching of a new squoze deck. The exceptions never reproduced are:

- a) Invalid operation codes, which are replaced in the listing by ///.
- b) Invalid symbols, such as those longer than six characters, which are replaced by ////.
- c) The shortened forms of extended operation codes which are changed and listed in their extended forms, e.g., the instruction WRS 1169 is listed as WTBB 1.

Also, words generated by the BCI, DEC, LBR and OCT instructions or by macro-instructions are not normally listed in detail. Instead, only a title line and the first word generated by these instructions are printed. However, these may be listed in detail if the pseudo-op DETAIL is used as previously defined.

When a squoze deck is listed, the comments are aligned with the first comment in the program and therefore may not be lined up exactly as in the source deck listing.

Symbolic listings show the job title, page number, and date in the upper right-hand corner of each page and are followed by 50 lines of printing. The listing itself consists of several parts.

The symbolic instructions for the program are listed, and octal equivalents are normally given. These instructions are assigned numbers from two reference systems (i.e., relative and alter numbers) as described below.

Appearing next, at the option of the user, is a listing of all defined symbols used in the program. These symbols appear five on each line, in alphabetic order. Multiply-defined symbols appear at the end of the table with the numbers of the pages on which they appear.

1.1.2.1 Reference Systems

The two numbering systems previously mentioned (<u>relative</u> and <u>alter</u>) are used to refer to words in a program. These numbers are assigned initially by the Compiler and are changed, if necessary, by Modify and Load when a new squoze deck is punched.

Relative Numbering

A relative number is an integer which indicates position of a machine word relative to a preceding word having a location symbol. The positions thus indicated are the relative positions of instructions the last time a squoze deck was punched.

Since relative numbers, in a sense, indicate storage locations occupied by machine words, they are assigned only to those instructions which, when loaded for execution, occupy locations. Thus, relative numbers are never assigned to principal pseudo-instructions (BES, BOOL, BSS, END, EQU, HEAD, ORG, SYN, TCD), generative pseudo-instructions (BCI, DEC, DUP, LBR, OCT), or programmer macro-instruction definitions.

Relative numbering begins when the first location symbol of a program is encountered. The word associated with this symbol is numbered 0 (although not shown on listings) and the next word is numbered +1. Numbering continues until either another word with a location symbol or an instruction with a principal pseudo-operation is encountered. When a new symbol is encountered, the process is started again. If, however, relative numbering is suspended by one of the pseudo-operations, it is not reinitiated until a new symbol is encountered.

Words for which a positive relative number cannot be computed are given a negative relative number, i.e., a number relative to a succeeding symbol, if that can be computed. If neither can be computed, no relative number is shown.

Although only one relative number is shown on the listing for a given word, there exists, in general, many other equivalent relative numbers, both positive and negative, any one of which may be used when referring to that word. For example, in the following list, the word numbered +1, relative to the symbol MASK, has the equivalent number +7, relative to RESTOR, or -1, relative to WRITE, etc.

82	RESTOR	AXT	**0 , 1	RESTOR
8 3	+1	\mathbf{AXT}	**0 , 2	INDEX REGISTERS
84	+2	\mathbf{AXT}	**0, 4	CONTENTS
85	+3	\mathbf{AXT}	2, 4	RETURN
86	+4	\mathtt{SLN}	1	TURN SENSE LIGHT 1 ON
87	+5	TRA	PRINT	
88	MASK	\mathbf{OCT}	373737373737	
89	+1	\mathbf{OCT}	377737773777	
90	WRITE	PZE	WKAREA,,24	
91	IMAGE	BSS	NUMBER, 0	
92	NUMBER	\mathbf{EQU}	24	
93	ZERO	EQU	0	
94	TSTBIT	PZE		STORAGE FOR TEST BIT
95		END	PRCOMM	

There is no number for a word relative to a symbol which is separated from that word by a principal pseudo-operation. For example, in the listing the words preceding the BSS with the location symbol IMAGE have no numbers relative to the symbol TSTBIT.

Alter Numbering

Alter numbers are numbers for the symbolic cards in a source program deck; they are assigned to all cards except:

- a) Those which contain ETC and MACRO instructions
- b) Those which define programmer macro-instructions
- c) The Modify and Load pseudo-instructions

Generative pseudo-instructions (such as BCI) and programmer macro-instructions are assigned alter numbers. The words generated by the instructions are not assigned numbers.

1.1.2.2 Sample Listing

The following sample presents the data found on an SOS symbolic listing:

- a) Storage locations in octal
- b) Octal equivalent of each instruction
- c) Alter numbers
- d) Symbolic locations
- e) Relative numbers
- f) Operation codes
- g) Variable field (containing the address, tag, and decrement portions and/or a comment section)**

a	b	С	d	е	f	g
	TRI FUN HLS 12/	01/6	1			Page 1 TRIG FUNCTION Problem
		2* 3*			ORG	HOMER SNIDER JOB VGPP, SIN, COS 15000
35230	0 77400 1 00132	5	XA		AXT	90, 1 Generate
35231	0 50000 0 35602	6	XA1		CLA	ZERO fixed-
35232	0 40000 0 35736	7		+ 1	ADD	ONEX \dispoint
35233	0 60100 1 35736	8	XA2	:	STO	FIXED-91,1 /numbers
35234	2 00001 1 35232	9	XA3		TIX	*-2, 1, 1 0 to 90
35235	0 77400 1 00022	10	XA4		AXT	18, 1 Generate
35236	0 50000 0 35737	11	XA5		CLA	ONEF \floating-
35237	0 30000 0 35737	12		+ 1		ONEF 'point
35240	0 60100 1 36010	13	XA6		STO	Float -18, 1 numbers
35241	2 00001 1 35237	14	XA7		TIX	*-2, 1, 1 (2 to 19
35242	0 50000 0 35737	15	XA8		CLA	ONEF Float one.
35243	0 60100 0 35740	16		+ 1	STO	E
35244	0 77400 2 01166	17		+ 2	AXT	630,2
35245	0 77400 1 00266	18		+ 3	AXT	182,1
					CORE	FIXED, X, 0, 0
35246	0 62500 0	19	CORE	1	STL	2169

^{**}The variable field section must start in column 16. The comment section must be separated from the end of a preceding variable field by at least one blank. In no case can it start to the left of column 17.

1.1.3 SOS MODIFY AND LOAD

The input to the SOS Compiler is a symbolic source program from which a compact binary-coded symbolic (squoze) program deck is produced. This deck contains all information supplied in the source program, including remarks cards and comments from instruction cards. Squoze decks produced by the Compiler may be used with symbolic decks as input to subsequent Compiler passes to produce one squoze output deck. Thus, a program can be written in parts and each part debugged before all are combined.

Squoze decks produced by the Compiler are also used as input to Modify and Load. Since all symbolic information is available, Modify and Load has three major advantages over previous assembly systems:

- a) Changes in symbolic form can be incorporated into the program by Modify and Load.
- b) Symbolic changes do not require the source deck to be reprocessed by the Compiler.
- c) Symbolic information is available and may be retained for printing during debugging runs, thus making debugging easier.

The main functions performed by Modify and Load are:

- a) Modification of a squoze program on the basis of symbolic information supplied with the squoze deck.
- b) Loading the modified version of a program into storage in preparation for execution of the program.

In addition to the above, Modify and Load also offers the following features:

- a) When desirable, a new squoze deck incorporating symbolic modifications may be prepared. (A new squoze deck is automatically prepared when a modification affects a heading card.) Generally, this option should be exercised when the number of modification cards is approximately equal to the number of cards in the squoze deck.
- b) A symbolic listing of a program can be prepared from a squoze deck which includes no modifications. (A new symbolic listing is automatically prepared when a new squoze deck is punched.)
- c) An absolute binary version of a program may be obtained from a squoze deck. Although this option is available to the user, little benefit is derived by exercising the option until a program has been completely debugged, since the debugging and modification features of SOS can only be used with squoze program decks.

1.1.3.1 Pseudo-Operations

The SCAT language includes five pseudo-operations by which changes may be made to a program at Modify and Load time. The use and effect of these pseudo-operations are described below.

To accomplish modifications, the modification instructions and any words to be inserted in a program are punched in symbolic form and used as input with the squoze deck. The changes indicated in these cards are made in the program before it is loaded into storage, but do not affect the squoze deck until a new deck is punched. At that time, the changes are physically incorporated in the new squoze deck.

The effects of the modification pseudo-operations, when loading a program into storage and when preparing a new squoze deck, are equivalent to and could be accomplished by making the required changes in the original symbolic source program, reprocessing with the Compiler, and then loading the new squoze deck. In the discussion that follows, only the effects which the pseudo-operations have on the squoze deck are indicated.

Throughout the discussion, each change is indicated as though it were the only one affecting the program, regardless of the actual number. That is, all changes must be indicated in terms of the current deck and the associated listing.

CHANGE

The CHANGE pseudo-operation can be used to delete words from a program, to insert additional words into a program, or to do both, depending on the form of the instruction. When CHANGE is used, modifications are specified in terms of relative numbers.

CHANGE instructions may be used to delete or insert words with which location symbols are associated, in which case the location symbol is also inserted or deleted. When a word which has a location symbol is deleted, the symbol is deleted from the dictionary and may, therefore, be used subsequently as a location symbol for another word. No location is required with CHANGE; if one is present it is ignored.

Two forms of the CHANGE instruction are permissible. The first is:

	1	Locat	ion		Operation			Address, Tag, Decrement/Count	
1	 	2	6	7	8	14	15	16	\setminus
			-		СНА	NGE		A+n, B+m	{

A + n and B + m represent relative expressions, i.e., A and B are symbols and m and n are integers which may be positive, negative or zero.

This form indicates that all words in location A+n to B+m, inclusive, are to be deleted from the program. If, in addition, symbolic instruction cards immediately follow an instruction in this form, the instruction also indicates that the words in the symbolic cards are to be inserted beginning with location A+n. Since insertions are made as in an assembly, the words following B+m are automatically adjusted and the number of insertions and deletions need not be equal.

When any, but not all, of the words generated by either BCI, DEC, LBR, MACRO, or OCT are deleted by a CHANGE, each of the subfields remaining from the original instruction is carried as a separate word and is assigned a separate alter number. In the listing, however, only the absolute word and relative and alter numbers are shown. No symbolic information is shown in the operation, variable, and comments fields. In all other changes to which a CHANGE can apply, the comments associated with deleted words are deleted from the squoze deck; remarks cards falling within the range of deletion by a CHANGE are not deleted from the program.

When a CHANGE instruction of the form shown above affects a headed area, it must be written:

	Loca	tion		Operation			Address, Tag, Decrement/Count	7
1	2	6	7	8	14	15	16)
	 			СН	ANGE		H\$A, H\$B+m	<u> </u>

where H represents a heading character.

The second form permitted is:

	Loca	Location		Operation			Address, Tag, Decrement/Count
1	2	6	7	8	14	15	16
				СНА	CHANGE		A+n

where A is a symbol and n is an integer which may be positive, negative, or zero.

This form of a CHANGE instruction indicates that the symbolic instruction cards which immediately follow it are to be inserted between the words in location A + n and A + n + 1. No deletions are made. If no symbolic cards follow an instruction in this form, the instruction is ignored.

When a generative pseudo-operation is inserted into a program by means of a CHANGE instruction, the individual terms are not assigned separate alter numbers.

When insertions are to be made in a headed area, the second form of the CHANGE instruction is written:

	Loc	ation		Oper	Operation		ration		Address, Tag, Decrement/Count	: (
1	2	6	7	8	14	15	16	\mathcal{L}		
				CHA	ANGE		K\$A+n	<u> </u>		

where K represents a heading character and A + n is as previously described.

In the following list of restrictions, all statements are made in terms of the headed forms of CHANGE. These restrictions can be applied to the unheaded forms by considering an unheaded symbol to be headed by a blank.

Restrictions:

- a) In a CHANGE instruction of the first form, H\$A + n must be either less than or equal to H\$B + m; otherwise, the CHANGE and the symbolic cards following it are ignored.
- b) No principal pseudo-operation (BES, BOOL, BSS, END, EQU, HEAD, ORG, SYN, TCD) may appear within the range of the symbols A to B + m.
- c) No principal pseudo-instruction, listing pseudo-instruction, or remarks card may appear as an insertion by means of a CHANGE. Any insertion which violates this restriction is ignored.
- d) Remarks cards and listing pseudo-operations cannot be deleted by a CHANGE. When remarks cards or pseudo-operations appear between H\$A + n and H\$B + m, inclusive, they are not affected by the CHANGE.
- e) No CHANGE instruction should specify the deletion of only part of the words generated by a VFD pseudo-operation.

- f) If a programmer macro-instruction is inserted by means of a CHANGE, the definition must also be included with the group of modifications. This does not mean that the definition must be included with the same CHANGE that is to insert the macro-instruction. Instead, it may be included by an ALTER or by another CHANGE. The definition may also be placed in front of the group of modifications and need not be preceded by a Modify and Load pseudo-operation.
- g) A modification by a CHANGE instruction must not overlap another modification by an ALTER (see below) or a CHANGE.

Example 1: Assume that the instructions with the alter numbers 79 and 80 in the following listing are indicated to be in error.

78	1	TRA	CLEAR + 4	RETURN
79	EXIT	\mathbf{AXT}	,1	
80	1	///	1	IF SENSE LIGHT 1 IS ON DO NOT
81*				RESTORE INDEX REGISTER 1

To remove the error indication by means of a CHANGE, the following instructions are necessary:

	Location			Operation			Address, Tag, Decrement/Count	
1	2	6	7	8	14	15	16	(
EXI	T			CHAN AXT SLT	IGE		EXIT, EXIT+1 **0, 1 1	7

(NOTE: **0 was arbitrarily selected to indicate modified addresses.)

Assuming there are no modifications which affected the alter numbering of previous instructions in the listing, the instructions corrected would appear in a listing of the modified deck as:

78	+1	TRA	CLEAR + 4	RETURN	
79	EXIT	\mathbf{AXT}	**0 , 1		
80	+1	SLT	1		
81*				DO NOT RESTORE IR 1	L

(The octal absolute has been omitted for the sake of clarity; however, the absolute equivalents would also be changed.)

Example 2: Assume that the instruction SLT 1 is to be inserted following the instruction which has an alter number 9 in the list below without deleting any instructions.

6	PRCOMM	CLA	1, 4	GET PRINTER CONTROL WORD
7	+1	\mathtt{TDL}	* 2	
8	+2	WPDA		DOUBLE SPACE PRINTED IF
9	+3	WPDA		CONTROL NEGATIVE, SINGLE IF+
10	+4	SXA	RESTOR, 1	SAVE INDEX
11	+5	SXA	RESTOR+1,2	REGISTER

The required modification cards are:

	Location			Operation			Address, Tag, Decrement/Count	7
1	2	6	7	8	14	15	16	(
				CHAI SLT	NGE		PRCOMM +3 1	{

After this change has been made, the listing appears as follows (assuming that there are no changes which affect previous instructions):

6	PRCOMM	CLA	1,4	GET PRINTER CONTROL WORD
7	+1	\mathtt{TPL}	* +2	
8	+2	WPDA		DOUBLE SPACE PRINTER IF
9	+3	WPDA		CONTROL NEGATIVE, SINGLE IF +
10	+4	SLT	1	
11	+5	SXA	RESTOR, 1	SAVE INDEX
12	+6	SXA	RESTOR $+1$, 2	REGISTER

ALTER

The ALTER pseudo-operation is analogous to CHANGE in that it may occur in two forms similar to those of CHANGE and may be used to make insertions, deletions, or both. ALTER, however, inserts and/or deletes the equivalents of symbolic source program cards instead of machine words.

There are two permissible forms for ALTER. The first is when \mathbf{N}_1 and \mathbf{N}_2 represent alter numbers.

		Loca	tion		Operation			Address, Tag, Decrement/Count	(
	1	2	6	7	8	14	15	16	(
I					AI	LTER		N ₁ , N ₂	(

This form indicates that the information corresponding to alter numbers $\rm N_1$ through $\rm N_2$, inclusive, is to be deleted from the program. If symbolic cards are associated with an ALTER instruction in this form, the instruction also indicates that the cards are to be inserted into the program between $\rm N_1$ -1 and $\rm N_2$ +1. As with CHANGE, the number of insertions need not be equal to the number of deletions since the words following $\rm N_2$ are automatically adjusted.

In the second form N is also an alter number:

	Location			Operation			Address, Tag, Decrement/Count
1	2	6	7	8	14	15	16
				AL	TER		N (

This form indicates than no deletions are to be made and that the associated program modification cards are to be inserted between the symbolic instructions numbered N and N+1.

Restrictions:

- a) For an ALTER instruction on the first form, N_1 must be less than or equal to N_2 ; otherwise, the instruction and the symbolic cards to be inserted are ignored.
- b) Remark cards and DETAIL, LIST, TITLE, and UNLIST pseudoinstructions cannot be deleted by an ALTER. When an ALTER specifies alter numbers which include one of these in their range, the ALTER does not affect the remarks cards or listing pseudoinstructions.
- c) An ALTER instruction cannot delete an END card without also inserting an END card.
- d) An ALTER cannot insert an END card without also deleting an END card. If an ALTER includes an END and does not specify the deletion of an END, the END to be inserted is ignored.

- e) If a programmer macro-instruction is inserted by an ALTER, the definition must also be included with the list of modifications. This does not mean, however, that the definition must be included with the same ALTER that is to insert the macro-instruction. Instead, it may be included by a CHANGE or by another ALTER. The definition may also be placed in front of the group of modifications and need not be preceded by a Modify and Load pseudo-instruction.
- f) A modification by an ALTER must not overlap a modification either by another ALTER or by a CHANGE.

Example 1: Assume that the instruction is to be corrected with alter number 5 in the following listing:

4*				
5x		ORG	START	
6	PRCOMM	CLA	1, 4	GET PRINTER CONTROL WORD

The instructions necessary to accomplish the corrections are:

	Loca	tion		Oper	ration		Address, Tag, Decrement/Count	7
1	2	6	7	8	14	15	16	2
				AL OR	TER G		5,5 3000	\int

After this correction has been incorporated and, assuming that there are no changes which affect the preceding remarks cards, the listing appears:

```
4*
5 ORG 3000
6 PRCOMM CLA 1, 4 GET PRINTER CONTROL WORD
```

Example 2: Assume that in the following listing the instructions with alter numbers 92 and 93 are to be deleted.

91	NUMBER	$\mathbf{E}\mathbf{Q}\mathbf{U}$	24	
92	NUMBER	EQU	12	
93	ZERO	\mathbf{EQU}	0	
94	TSTBIT	PZE		STORAGE FOR TEST BIT

The required instruction is:

	Loca	tion		Operation			Address, Tag, Decrement/Count	7
1	2	6	7	8	14	15	16	1
				AL	TER		92, 93	ζ

After this change is made the listing appears (assuming no modifications affecting preceding instructions) as:

91 NUMBER EQU 24 92 TSTBIT PZE STOR

STORAGE FOR TEST BIT

SYMBOL

The SYMBOL instruction permits the assignment of a location symbol to a word without requiring the deletion and subsequent insertion of the word.

There is one form of a SYMBOL instruction:

Γ		Locat	Location Operation Address, Tag, Decrement/Co		Operation		Address, Tag, Decrement/Count	7	
	1	2	6	7	8	14	15	16	\
		В			SYMBOL			A + n	~

B represents a symbol of from one to six characters which is to become associated with the word previously assigned the relative location expression A + n (use relative numbers only).

If SYMBOL is used to associate a location symbol with a word which already has a location symbol, the new symbol does not replace the old; instead, the two are made synonymous by an EQU instruction. However, if the symbol in the location field of the SYMBOL instruction has been previously defined in the program, it is defined again with the new value and becomes a doubly-defined symbol.

If the location field or the variable field of a SYMBOL instruction is blank, the instruction is ignored.

When a SYMBOL instruction is to assign a symbol to a word in a headed area (for example, when A is headed) the instruction is written:

	Loca	ation		Operation			Address, Tag, Decrement/Count		
1	2	6	7	8	14	15	16	1	
	H	3		SYMBOL			H\$A + n	3	

H is the character by which A is headed; B and A + n are as described previously.

Restrictions: If a principal pseudo-operation appears in the range H\$A and H\$A + n, inclusive (or if A is unheaded), the SYMBOL pseudo-instruction above has no effect on the program.

Example: Assume that a symbol must, for convenience, be assigned the instruction with alter number 25 in the following listing:

16	CON6	PDX	6, 2	
17	+1	LGR	18	COMPUTE # INSERT WORDS -
18	+ 2	ADD	1, 4	START ADDRESS AND
19X	+3	STO	/////	STORE.
20	+4	CLA	CON6	INITIALIZE FOR OCTAL IF TAG
21	+ 5	TQP	* + 2	OF PRINT CONTROL IS 4.
22	+6	ARS	1	IF OUTPUT IS OCTAL STORE
23	+7	STA	STRTWD-2	3 IN CONVERSION ADDRESS
24	+8	CLA	SWITCH	
25	+9	LLS	0	
26	+ 10	STO	SWITCH	
27	+ 11	AXT	24, 1	
28	+12	TCOA	*	DELAY UNTIL CHANNEL AVAILABLE
29X	+ 13	NOP	WKAREA+23,1	
30X	+ 14	STZ	WKAREA+24,1	CLEAR WORK AREA FOR
31	CLEAR	TIX	*+1,1,1	CONVERSION

	Loca	tion	on Operation		Operation		Address, Tag, Decrement/Count	7
1	2	6	7	8	14	15	16	7
	SHI	SHIFT		SYMBOL			CON 6+9	

The symbol instruction above would appear in a subsequent listing (assuming no other changes) as:

16	CON6	PDX	6, 2	COMPUTE # INSERT WORDS -
17	+ 1	LGR	18	START ADDRESS AND
18	+2	ADD	1,4	STORE.
19X	+3	STO	/////	INITIALIZE FOR OCTAL IF TAG
20	+4	CLA	CON6	OF PRINT CONTROL IS 4.
21	+ 5	TQP	*+2	IF OUTPUT IS OCTAL STORE
22	+6	ARS	1	3 IN CONVERSION ADDRESS
23	+7	STA	STRTWS-2	
24	+8	CLA	SWITCH	
25	SHIFT	LLS	0	

26	+1	STO	SWITCH	
27	+2	AXT	24,1	
28	+3	TCOA	*	DELAY UNTIL CHANNEL AVAILABLE
			WKAREA+23,1	
30X	+ 5	STZ	WKAREA+24,1	CLEAR WORK AREA
				FOR CONVERSION

ASSIGN

The ASSIGN pseudo-instruction defines or redefines symbols by insertion of EQU, SYN or BOOL cards. The form of an ASSIGN instruction is illustrated below:

	Location			Operation			Address, Tag, Decrement/Count	7
1	2	6	7	8	14	15	16	
				ASS	IGN		Н	3

where H represents a heading character which may be a blank.

This instruction must be followed by at least one EQU, SYN or BOOL instruction to perform one of the following functions:

- a) To define new symbols and undefined symbols in a program.
- b) To redefine symbols originally defined in a program by EQU, SYN or BOOL instructions.

An ASSIGN instruction may not be followed immediately by any instruction other than EQU, SYN, BOOL or SYMBOL. (Note that a SYMBOL following an ASSIGN does not terminate the effect of the ASSIGN.)

If an ASSIGN instruction specifies a non-blank heading character, all the symbols used in the following EQU, SYN, and BOOL instructions are headed by that character. (In addition, the only EQU, SYN, and BOOL instructions processed are those for which the location symbol has been previously defined by an EQU, SYN or BOOL card and is not multiply defined. Under these conditions, the new definition replaces the old one.)

When an ASSIGN specifies a blank heading character, the EQU, SYN, and BOOL instructions are treated as follows:

a) If the symbol in the location field of a SYN, EQU, or BOOL instruction is undefined or is new to the program, the symbol becomes defined as usual. The EQU, SYN, or BOOL instruction defining the symbol is

inserted at the beginning of the program, preceded only by remarks included at the beginning of the source program deck.

- b) If the symbol in the location field of a SYN, EQU, or BOOL instruction is defined in the new program by a SYN, EQU, or BOOL and is not multiply defined, the new definition replaces the old one at the same point in the program.
- c) In all other cases, the symbol in the location field of a SYN, EQU or BOOL instruction is multiply defined in the program.

When a SYMBOL card follows an ASSIGN, the location symbol is headed by the heading character of the ASSIGN, if the location symbol is less than six characters long.

The symbol in the variable field of the SYMBOL is also considered headed under the same condition.

Example: Assume that the symbols WKAREA and IMAGE are to be equated in a program. The instructions necessary are:

Location	Operation	Address, Tag, Decrement/Count	5
WKAREA	ASSIGN SYN	IMAGE	

The listing might then appear as follows (assuming there are no modifications which affect the four remarks cards at the beginning of the source program):

4*			
5	WKAREA	$\mathbf{E}\mathbf{Q}\mathbf{U}$	IMAGE
6X		ORG	START

1.1.4 SOS DEBUGGING MACROS

The principal function of debugging macros is to permit the programmer to investigate the contents of storage or control panel during the execution of his program. The debugging macros are used during the development phases of a program and are removed after debugging has been completed.

The debugging macros can be thought of as extensions of the pseudooperations of the SCAT source language and, as such, can be inserted into the program either while coding or as modifications during Modify and Load. In the latter case, use of the ALTER pseudo-operation inserts the debugging macro at the desired location in the program. When the Compiler or Modify and Load interprets a debugging macro, a TXL branch occurs to the debugging subroutine, the function called for is executed, all main program indicators and storage cells are restored, and program control returns to the main program at the instruction immediately following the debugging macro.

Through the use of debugging macros, the programmer has the option of specifying the format of information to be printed and if the data is to be printed on-line or off-line.

There are several categories of debugging macros: information macros, which request information output; modal macros, which specify the format of the output; and conditional macros, which permit output selectivity.

1.1.4.1 Variable Fields

Since the function of debugging macros is varied, the variable field of the macros is adapted to the function of each. In general, the variable fields contain three types of information: location, format and count:

- a) Location—specifies the proper area of activity for core storage cells and the index.
- b) Format—indicates the format required for output results of information macros; specified format of VFD-introduced blocks of storage.
- c) Count Specifies:
 - 1) Position of binary point in fixed-point format
 - 2) Number of times a conditional macro is satisfied or unsatisfied
 - 3) Increment of every conditional macro

A typical debugging macro may be:

Location	Operation	Variable Field	
	CORE	L ₁ , L ₂ , F, IT ₁ , IT ₂	

 L_1 = first location (absolute.or symbolic)

 L_2 = second location (absolute or symbolic)

F = this field designates the format of the output and may be coded as follows:

<u>Format</u>	Code
Symbolic instruction	S
Fixed-point number	X
Floating-point number	\mathbf{F}
Octal integer	О
Hollerith BCD information	H
Variable	V

 ${\rm IT}_1$ (or ${\rm T}_1{\rm I}$)—Indirect addressing and the tag information for the first location.

 ${\rm IT}_2$ (or ${\rm T}_2{\rm I}$)—Indirect addressing and the tag information for the second location.

1.1.4.2 Information Macros

a) CORE L₁, L₂, F, IT₁, IT₂-execution of this macro reads out the core memory block from the lower location (defined by L₁ and IT₁) to the upper location (defined by L₂ and IT₂) in the specified format.

If the effective upper location is zero, the block of core memory from the lower effective location to the top of core memory is read out. If the effective upper location is nonzero, it must not be less than the effective lower location.

- b) PANEL (no variable field)—execution of this macro reads out the:
 - 1) Accumulator and MQ-each in both octal and floating-point format.
 - 2) Index Registers 1, 2 and 4-each in both octal and decimal.
 - 3) Sense Indicators—as an octal number whose binary equivalent has 0's for indicators which are off and 1's for those which are on.
 - 4) Sense Lights and Sense Switches—each a binary number, with 0's for those which are off or up, and 1's for those which are on or down, respectively.
 - 5) Entry Keys—as an octal number.

6) Accumulator Overflow, Divide Check, and Input/Output Check Indicators—each either on or off.

1.1.4.3 Modal Macros

There are three modal macros with variable fields (USE, FORMAT, and POINT) and three without (ON, OFF, and NUCASE). These macros set modes for subsequently executed debugging macros.

a) USE A₁, A₂, A₃, ..., A_n—in large systems particular memory locations may be occupied by different programs at different times. Consequently, if a programmer wants to read out certain locations to debug his program, he may get the location symbols of a program other than his. This situation becomes apparent when the CORE macro is used, since it scans the lists of ORG's in the Dictionary table, —from the last entry to the first—until it finds the first program containing the locations sought. For example, suppose the following system of programs is compiled using ORG to designate the origin of each program:

ORG	3584	3584-3996	(a)
ORG	4033	4033—5025	(b)
TCD	4650	4420-32000	(c)
ORG	4420	3000-3371	(d)
ORG	3000	30-3520	(e)
ORG	30	4033—8986	(f)
TCD	12785		
ORG	4033		
END	12785		

When an ORG pseudo-instruction occurs after the start of the program and causes the programs to be compiled with common locations, the USE macro can be used. In the example, b, c, and f have common locations as do d and e. If the programmer desired the contents of location 4054 in program b, the instructions or commands referring to that location (in symbolic or command format), and only a CORE pseudo-instruction is used, CORE would: (a) scan the list of ORG's, starting with the last entry f, (b) find that program f uses location 4054, and (c) dump the symbols from program f instead of b.

DECIMAL LOCATIONS	LOADING PHASE	FIRST EXECUTION PHASE	SECOND EXECUTION PHASE
0			
1000			
2000		e	
3000			
4000	•		•
5000			A STATE OF THE STA
6000			
7000			
8000			
9000		e	gad.
10000			
31000			
32000			

USE reorders the ORG list so that the table is searched in the order indicated by the symbols in its variable field. The variable field should contain at least one location symbol (any one) which is unique to the program to be dumped. Therefore, if B is a location symbol in program b, by using USE B, A, CORE would find b 4033-5025 as the last entry and read out the correct information.

To give a more complete example, suppose that programs a and b are used initially to call in, from tape, programs c, d, and e which constitute a "first execution phase." After this, program f is brought in from tape and used with program d during a "second execution phase" (see illustration). For this example, let A, B, C, ..., F be location symbols associated with programs a through f respectively; let location symbol F1 = 5020, C1 = 5021 (in programs f and c respectively); let

location F2 = 4050; and let location 4054 contain "TRA 5022." Then by using "USE C, E" prior to any output from the first phase and "USE F, D" prior to any output from the second phase, "CORE 4054, 4054, S" given during the first phase will read out "F 2 + 4 TRA C1 + 1". However, if this CORE macro is given during the second phase, it will read out "F2 + 4 TRA F1 + 2."

It is essential that the correct span governed by each ORG be recorded in the Dictionary by associating a symbol with the last location prior to each ORG, TCD, and END. If this is not done, the symbols of higher origined programs (higher order core locations) or negative absolute location numbers (rather than symbollic relative) will be associated with the desired locations if they lie outside any recorded span.

If the USE macro is not used, or the desired location is outside the section designated in the USE variable field, the ORG with the highest numerical core location lower than the numeric address of the designated program will be assumed to be the effective ORG. If symbols have been assigned to define the span of each ORG and no USE macro is used, the following symbols will be attached to all outputs regardless of which phase is being executed at the time the section is read out.

Decimal Locations	Symbols From Program
30-2999	(e)
2000—3583	(d)
3584—4032	(a)
4033-4419	(f)
4420-32000	(c)

Therefore, since programs a, c, and d each lie completely within one of the above regions, correct symbols would be attached to dumps of locations within those programs while programs e and f, which are interrupted by origins of other programs, will not be dumped correctly. Note that two or more sections origining at the same core location, without a USE, will cause the symbols of the section physically appearing last in the compilation to be employed. Thus, symbols of program b could never be seen unless USE is given.

Since USE changes the original order of the Dictionary search and remains effective until it is nullified by another USE, a "USE E" during phase 1 in the example would dump the wrong locations during phase 2 unless a "USE D" were given. This is true even though program d origins at a numerically higher location than program e and

would have been read out correctly in phase 2 if no USE had been employed earlier.

The USE macro is assigned an alter number, appears on a compilation listing, and occupies $(2n + 3)_{10}$ locations where n is the number of symbols specified in the variable field.

b) FORMAT B₁, F₁, B₂, F₂,..., B_n, F_n—this macro defines the format code V. Words which have been compiled by a VFD pseudo-instruction, or otherwise involve hetrogeneous format, are read out by information macros which stipulate format code V. Therefore, V must be defined by prior execution of the correct FORMAT macro. For example, if locations B through B + 2 contain the octal words 004003040010, 764240000000, and 254560000000, respectively, the macros

cause the first (leftmost) six bits of location B to be output as an octal number, the next 15 bits as a fixed-point number, the next 25 bits (containing to location B+1) as a fixed-point number, and the next 12 bits as Hollerith information. For Hollerith, the number of bits should be a multiple of six.

c) POINT N—this macro defines the position of the binary point within a word to be read out in fixed-point format.

N is an integer from 0 to 35 which indicates the number of bits which lie to the left of the binary point.

- d) ON (no variable field)—this macro, prior to execution of OFF or NUCASE, prints debugging information on-line and writes it on the BCD output tape for peripheral printing.
- e) OFF (no variable field)—this macro prior to execution of ON, writes debugging information on the BCD output tape for off-line printing. This is the normal condition which prevails prior to execution of any ON macro and after NUCASE.
- f) NUCASE (no variable field)—if a program remains in core memory while it is repeatedly executed for different cases, as for example when new data cards are read into a fixed area of core memory, the NUCASE macro is used at the start of each case to reset the POINT and ON modal macros to normal. It also resets all counts generated by count type conditional macros to zero and reads out a case identification number.

1.1.4.4 Conditional Macros

- a) WHEN—when the variable field conditions are satisfied, subsequent information macros will be executed.
- b) UNLESS—when the variable field conditions are not satisfied, subsequent information macros will be executed.
- c) AND—this macro connects conditional and information macros and extends the power of the WHEN and UNLESS macros. For example, it may be difficult to specify both upper and lower limits of a given variable with one WHEN macro. However, if one limit is specified by a leading WHEN macro, the other limit can be specified by a following AND macro. When using the AND macro with a WHEN macro, both conditions must be satisfied.
- d) OR—this macro connects a conditional and an information macro and extends the power of the WHEN and UNLESS macros. Unlike the AND macro, the OR macro permits the specification of more than one condition, any one of which permits the execution of subsequent information macros.
- e) EVERY N-this macro specified the output increment for successive passes through a program loop. Its variable field consists of an integer N which allows a succeeding information macro to be executed the first time and, subsequently, every Nth pass through a program loop.
- f) Variable Fields of Conditional Macros—all conditional macros except EVERY can use the following general format for their variable fields:

$$L_1, R, L_2, IT_1, IT_2$$

The relation subfield R is coded in one of the following ways:

R Code	Meaning	Comparison Employed by (DB)
L E G LL LE	Less than Equals Greater than Logically less than Logically equals	CAS CAS CAS LAS LAS LAS (redundant to E)
LG	Logically greater than	LAS

The other subfields involve some additional conventions peculiar to the conditional macros.

The rules governing use of L_1 , L_2 , IT_1 and IT_2 are similar but not identical to those governing the CORE macro. The following rules govern the use of terms in the variable field (assume that OP is a WHEN, OR, UNLESS, or AND macro):

- 1) Case 1; OP (L₁):
 - (a) When L_1 is zero or blank, the macro is meaningless.
 - (b) When L₁ is 1 through 7, the contents of XR 1, 2 or 4 or their result is specified.
 - (c) When L_1 is greater than 7, a core storage cell is specified.

(NOTE: The above explanation is true if and only if at least one other subfield in the variable field is expressed. Otherwise, this form of the macro expresses a count type condition, for example, WHEN N, where N may be any number. See programming examples d, e and f.)

2) Case 2; OP L, R:

R expresses a relationship between L_1 and the term that follows, e.g., L_1 , E, L_2 (L_1 equals L_2).

- (a) All rules of Case 1 for L, apply.
- (b) R must be one of the six symbols established for the desired relationship.
- (c) Since L_2 is not expressed, the relationship specified is between L_1 and zero.
- 3) Case 3; OP L_1 , R, L_2 :
 - (a) All rules of Case 1 apply to both L_1 and L_2 .
 - (b) R must be one of the accepted six symbols.
- 4) Case 4; OP L₁, R, L₂, IT₁:
 - (a) If I is written anywhere in the fourth term, L₁ is indirectly addressed.
 - (b) L_1 is not inferred as an XR but is always a core storage location, regardless of magnitude.

- (c) If L_1 is blank, it is regarded as a tagged address of zero.
- 5) Case 5; OP, L_1 , R, L_2 , IT_1 , IT_2 :

All rules governing \mathbf{L}_1 apply to both \mathbf{L}_1 and \mathbf{L}_2 using \mathbf{IT}_2 as address modification specification for \mathbf{L}_2 .

1.1.4.5 Programming Examples of Debugging Macros

a) A STO X CORE B CLA Y

After execution of STO X, all of core storage is read out on BCD tape in addition to the panel information and, immediately following, control is returned to CLA Y.

b) Given: L50 is the location of 50

WHEN 4,G, L50 CORE

If the contents of XR4 is equal to or less than L50, core memory is not read out; only when the contents of XR4 is greater than 50 is core memory read out.

c) Given: Location 4 contains PZE 888; XR 2 equals 1

UNLESS 4, L, 2, I OFF

If the contents of LOC 888 is greater than zero, the subsequent output is off-line.

d) WHEN 8 (count type condition) CORE 800, 800, X

If this pair is inserted in a program loop for the first seven executions of WHEN, CORE is inoperative; following the eighth execution of WHEN, CORE becomes operative.

e) UNLESS 8 (count type condition)
POINT 18
CORE 800, 800, X

If this sequence has been inserted in a program, the number of location 800 is a fixed-point integer, properly read out until eight outputs have occurred. Thereafter, the sequence is inoperative.

f) WHEN 3 UNLESS 3 CORE A, A (count type condition) (count type condition)

If this sequence has been inserted in a program loop, the CORE macro becomes inoperative on the first and second passes, memory is read out on the third, fourth and fifth passes, and all subsequent passes are inoperative.

g) Given: X is to be read out when it lies between 50 and 70. 50,X and 70 are located at L50, LX and L70, respectively.

The macro program to give proper output can be written:

WHEN L50, L, LX AND LX, L, L70 CORE LX, LX, X

h) Using the given locations in example "g" to output X when it is less than 50 or greater than 70, the following macro is used:

UNLESS L50, L, LX AND LX, L, L70 CORE LX, LX, X

i) Given: X is to be read out when $X^2 \ge 100.X, 10$ and -10 are located at LX, L10 and LM10, respectively. The coding is:

WHEN LX, L, LM10 OR LX, G, L10 CORE LX, LX, X

j) Given: The first 50 non-negative values of X in a loop are to be read out. The coding is:

UNLESS LX, L, 0

OR 50

(count type condition)

CORE LX, LX, X

Here the UNLESS macro is associated with a count of outputs.

k) Given: X,Y and Z are located at LX, LY and LZ, respectively. X is to be read out the first 50 times any of the following three conditions are satisfied:

X exceeds Y and XR 4

X exceeds XR2

X exceeds Z and XR 4

The coding is:

WHEN LX,G,LY
OR LX,G,LZ
AND LX,G,4
OR LX,G,2
UNLESS 50
CORE LX,LX,X

(For a detailed explanation of the associative and commutative laws governing this type of sequence see page 29, Part 3, of the Share 709 SOS Manual.)

1.1.5 SOS MONITOR

The Monitor is a supervisory program written to control the processing of job decks through the computer. A job deck consists of a program deck and its associated control cards which designate the operation to be performed.

The control cards direct the Monitor to perform any or all of the following:

- a) Compile a program (listing and squoze deck as output).
- b) Modify and load a squoze deck for execution.
- c) Modify and punch a squoze deck to punch a clean (no modifications) squoze deck.
- d) Produce a listing of a squoze deck with or without modifications.
- e) Permit the use of debugging macros.

1.1.5.1 System Operation: Input Deck

When using the SOS system for an assembly, a debugging run, or an execution run, the first card of each job deck is a JOB card. The alphabetic characters J, O, and B are punched in columns 8-10 of the card. Also punched in columns 16-27 of the card are the name of the program and the programmer's name or initials to enable the operator to separate and return the results.

Immediately following the JOB card, a DATE card may be inserted. (DATE punched in columns 8-11 and six Hollerith digits in columns 16-21 for the month - day - year.) The DATE card will override a date entered in the MQ-keys on the console. The date entered either via the keys or a DATE card will appear on

every page of a compilation or punch-squoze listing. It will also appear on the first page of the output for all types of jobs.

Cards with columns 8-13 blank, placed between the JOB and succeeding cards, will be printed both on-line and off-line. The variable field of these cards might contain instructions to the operator or other remarks.

The input deck consists of any sequence of job decks, followed by a card punched PAUSE in columns 8-12. Job decks include the following possible categories:

a) Compilation Job Decks

- 1) Card punched JOB in columns 8-10 with the name of the program and the programmer (or his initials) in columns 16-27. Columns 11-15 must be blank.
- 2) Card punched CPL in columns 8-10 for column binary, or CPLRB in columns 8-12 for row binary output.
- 3) At least two remark cards, one with the name of the program and one with the name of the programmer.
- 4) Symbolic program deck from ORG to END card.
- 5) Blank card
- 6) PAUSE card

Non-modified, column or row binary squoze decks may be inserted in the symbolic deck if preceded immediately by a SQZ symbolic card. For column binary, SQZ is punched in columns 8-10; for row binary, SQZbRB is punched in columns 8-13. The squoze decks incorporated in the symbolic deck must be complete.

b) LS—List Job Deck

- 1) JOB card (as in a, 1 above)
- 2) Cards punched LS in columns 8 and 9
- 3) Squoze deck without modification
- 4) Blank card
- 5) PAUSE card

- c) Execution Job Deck
 - 1) JOB card
 - 2) Card punched LG in columns 8 and 9
 - 3) Squoze deck*
 - 4) Blank card
 - 5) Any number of data sentence decks**
 - 6) Card punched GO in columns 8 and 9
 - 7) PAUSE card

The card sequence with a squoze deck is of major importance; manual rearranging should be avoided. When no modification is desired, there is no change required in the squoze deck; it should be fed into the card reader exactly as produced in the card punch.

d) List Squoze Deck

This job deck gives a dump-type listing of the squoze deck with modifications. The listing does not contain comments. It looks like a dump using the CORE macro with the symbolic format specified.

- 1) JOB card
- 2) Card punched LG in columns 8 and 9

Original Squoze

Modification Deck

Miscellaneous cards preceding blank

1. Card punched MOD in columns 8-10

2. Modification cards

3. Card punched ENDMOD in columns 8-13

Remainder of squoze deck

- 1. Card punched DS1 in columns 8-10
- 2. Data sentence decks
- 3. Blank card

Data sentence decks may be used to provide for input data during the debugging of programs. Additional information concerning DS1 cards is found in subsection 1.1.5.3.

^{*} If modifications are to be added, they are to be inserted as shown below:

^{**} Data sentence decks are composed as follows:

- 3) Squoze deck with modifications
- 4) Blank card
- 5) Card punched LIST in columns 8-11
- 6) PAUSE card

This type of deck must be used if there are modifications.

- e) PS-Punch New Squoze Deck
 - 1) JOB card
 - 2) Card punched PS in columns 8 and 9
 - 3) Squoze deck with modifications
 - 4) Blank card
 - 5) PAUSE card
- f) PA-Punch Absolute Binary

The following job deck causes the squoze deck to be decoded and absolute binary cards to be punched according to SOS format.

- 1) JOB card
- 2) Cards punched PA in columns 8 and 9
- 3) Squoze deck with or without modifications
- 4) Blank card
- 5) PAUSE card
- g) Compile and Execute
- h) Punch New Squoze Deck and Execute
- i) List Squoze and Execute Deck
- NOTE: If the input squoze deck is in row binary for an LG, PA, PS or LS run, "RB" must be punched in columns 16-17 of the LG, PA, PS or LS card. (Leave these columns blank for a columnar binary input deck.) If the output deck for a PS or PA run is to be row binary, "RB" must be punched in columns 18-19 of the PS or PA card. (Leave columns blank for a columnar binary output deck.)

1.1.5.2 Effect of Control Card

Control Card	System Action Caused	Visible Results
JOB	Initializes the Monitor and causes it to read next card	Prints JOB and remarks from JOB card variable field on-line
CPLRB	Calls in the Compiler and transfers control to the Compiler. The Compiler compiles the program, gives an error list and punches a squoze deck. Control is then transferred to the Monitor, which reads in Modify and Load to obtain a Modify and Load error list and a program listing	Prints CPLRB on-line and off-line. Pass SYSTAP to C1 and C2. When C2 is in, the system tape rewinds. Prints error list on-line or off-line; punches squoze on-line or off-line in row binary
CPL	Same as for CPLRB	Same as for CPLRB, ex- cept the squoze deck is punched in column binary
PS	Calls in Modify and Load, punches a new squoze deck and gives a program listing. May be used with or without modifications. MOD and END MOD cards must be used even if no modifications are present	Prints PS on-line and off- line. Punches new squoze on-line or off-line. Gives new program listing on- line or off-line
LS	Calls in Modify and Load and gives a listing. No modifications are permitted	Prints LS on-line and off- line. Gives program list- ing on-line or off-line
LG	Calls in Modify and Load, transfers control to Modify and Load, decodes squoze and writes absolute program on B1. At end of loading, it transfers control to Monitor to read next control card. Modifications are permitted	Prints LG on-line and off- line

Control Card	System Action Caused	Visible Results
PA	Calls in Modify and Load, decodes squoze and writes absolute program on B1; then punches absolute binary. Mods are permitted	Prints PA on-line and off- line. Punches absolute
GO	Reads SNAP (the DB1 program) into core memory below 5670g; clears memory from 5670g to 0; loads program from B1 until a transfer card record is read; then transfers control to object card program	Prints GO on-line and off- line
LIST	Reads SNAP (DB1) into core memory below 5670 ₈ ; clears memory from 5670 ₈ to 0; loads program from B1 until a transfer card record is read; executes core dump from 5670 ₈ to 0; then transfers control to Monitor to read next control card	Prints LIST on-line and off-line
PAUSE	Halts Monitor and allows the program to continue without rewinding all tapes. Press START to read next control card	Prints PAUSE on-line and off-line
The Mercury v the control car tapes and halts	ersion of SOS does not permit use of d STOP, which in the standard SOS a the computer.	f another SOS control card, system rewinds all system

1.1.5.3 Specifications of the Data Sentence Program

A data sentence is defined to include an absolute decimal location giving the initial loading address; this is terminated by an equal sign (=) which is followed by the data. Consecutive words of data are separated by commas, and the end of the sentence is indicated by the marker (*); for example, 7083 = -52, 32. 1E5,39.1B6* is a data sentence which loads three numbers—integer, floating and fixed numbers—into location 7083 and the two locations following.

The normal sentence data is floating-point data, fixed-point data and decimal integers which are expressed according to regular SCAT rules and which may follow each other arbitrarily.

To introduce octal data, the letter O is punched with the octal numbers enclosed within parentheses; for example, 7083 = -52,32. 1E5,39. O(-7,7263), 509E20*. This sentence loads three decimals, two octals, and one decimal beginning at 7083.

The remaining rules of syntax are:

- a) The card is used from column 1 to column 72; punching is continuous.
- b) A sentence may start in any card column and extend to the end-ofsentence marker. It may extend beyond a card; more than one sentence may appear on a card.
- c) Punching on a card must end with a comma or with an end-of-sentence marker. If a blank then follows, the remainder of the card is ignored.
- d) The last sentence of a data block must end with a (\$) instead of (*) and should be followed by a symbolic expression. Transfer to this location is made after loading the data block; for example:

Card 1-A =
$$7192 = 5.1E3, 60.12, 301.2*$$

Card 2-B = $7195 = 70.1, O(-77), 70, 1B7C

These two cards comprise a data block which loads as specified and transfers to location C.

Two types of errors may occur during conversion:

- a) Overflow/Underflow-normal zero is stored; conversion of next field continues.
- b) Mispunch—when an illegal character is encountered, normal zero is stored and processing is continued for the next field.

Error messages indicating column number and record are given.

If either TCD's or DS1's are used, the program must anticipate the logical record arrangement and call program and data blocks after logical record 1 from tape into core memory by use of calling sequences of the form:

TSX 82, 4 PZE A,, B Bad data return

A is the number of the desired logical record (program file). A=0 means to read the logical record with the number that is one greater than the last one read. A nonzero means B is the location to which the Monitor returns after reading. B=0 causes return to the location specified by the TCD, END, or \$ card. (Each TCD and END card terminates one program file. The first such file is number 1.)

1.2 SOS MODIFIED FOR MERCURY

1.2.1 MERCURY CHANGES TO SOS

1.2.1.1 Monitor

The Mercury SOS System tape contains the "32K New York IBMonitor," with the following changes:

- a) The Mercury SOS Monitor initializes core storage locations above 3000₁₀ to zeroes at the start of each job. Share SOS initializes by inserting an STR instruction (operation code: -1000₈) in each location above 3000₁₀.
- b) For floating-point overflow and underflow, SOS Mercury prints off-line the location of the instruction which causes the overflow or underflow. SOS Mercury overflow sets bits 1 through 35 of the responsible register (AC or MQ) to 1's, but the sign remains unchanged and the program continues. Share dumps when an overflow occurs. For both Share and Mercury, underflow causes the responsible register (AC or MQ) to be cleared, i.e., set to + 0, and the program continues.
- c) A program may be terminated by transferring to SYSTEM or SYSERR without defining these symbols in the program. This obviates the necessity of returning to the SOS Monitor with a TRA 10 or 14, 110 or 114, or any other absolute location subject to change. SYSTEM gets the snaps taken by the job and then initializes for the next job. SYSERR gives a symbolic dump (in the format from the compilation listing) from SYSORG to the top of memory and then transfers to SYSTEM. A third symbol, SYSTRA, which immediately initializes for the next job, can be used.
- d) Three other SOS system symbols are available; these are primarily used for interjob communication in the dual-compilation scheme. After a job assumes control at execution time, it can change the contents of these locations to accomplish a change in the way SOS processes the next job.

Symbol	Normal Octal Contents	Purpose
SYSORG	5670	The origin value assigned by SOS to each program file unless the program specifies another by means of an ORG card. Also, the starting location of all core dumps taken by the system

Symbol	Normal Octal Contents	Purpose
SYSMIT	2201	Mediary Input Tape. Contains the program in binary and various SOS information such as the dictionary of program symbols
SYSMOT	2202	Mediary Output Tape. Contains the debugging information taken during execution, in binary

- SOS Library Tape appears in the Mercury SOS Monitor. The Share Monitor provides for these items but does not include them since they are functions of each installation.
- f) An on-line punching operation (CPL, PS, PA) will cause a JOB card to be punched on-line ahead of the deck. The off-line punching operations (CPL, PS, or PA), with SSW#6 up, will not punch a JOB card ahead of the deck on SYSPPT.
- g) To permit the above changes to be made without affecting the correspondence of alter numbers and locations between the Mercury and Share systems, some instructions have been moved and some remarks inserted.
- h) The PAUSE halt is 1738.
- i) Additional SOS symbols available in Monitor are:

Program Symbol	Comments
SYS2PR	Prints message on-line and on A2. TSX SYS2PR PZE A,, B where A is the first location of the BCI message and B is the number of words in the message (maximum 12 for on-line)
SYSBA T	Decrement receives C tape drive number for a batched copy of B1
SYSCPL	Area in monitor not used during LG
SYSFGO	Area modified to stack edited dictionaries
SYSFIL	Transfer location to space to dictionary on B1.

Program Symbol	Comments
SYSFLO	(See commentary on floating origin)
SYSPRM	Subroutine to reset memory TSX SYSPRM,4 PZE A,,B Reset word Memory from A to B inclusive is reset to contents of Reset word.
SYSPRT	Subroutine to print a line or write a BCD record on A2. TSX SYSPRT,4 PFX A,,B Redundancy return EOT Return PFX is PZE for A2, MZE for printer. A is the location of the BCD characters. B is the number of words (maximum of 12 for printer)
SYSTLD	SOS Tape Loader (A1) TSX SYSTLD, 4 PFX A,, T If PFX is PZE, tape is loaded; MZE, tape is positioned in front of first record of requested file. A is communication cell containing call number T is the return address. If T is zero, return is specified by loaded file
SYSXCD	Continues SOS without picking up snaps
⁶⁶ 10	Contains SOS tape number (normally 12018)
6710	Contains Library tape number (normally 12048)
⁷⁰ 10	Contains input tape number (normally 12038)
⁷¹ 10	Contains output tape number (normally 12028)
⁷² 10	Contains squoze or binary tape number (normally 1205 ₈).

Program Symbol	Comments
7610	Redundancy correction routine. Attempts to write 25 times or read 10 times before it stops running TSX 76,4 PFX A,,D Error return A is the address of first I/O command in string. D is the unit (such as 1206 or 1226 for A6 in BCD or binary, respectively) XR2 contains the record count PFX is PON for read, PTW for write end of file, PTH for write
⁸² 10	Mediary input tape (B1 Job Tape) loader TSX 82,4 PZE A,, B Bad data return where A is file call number; if zero, next file is loaded B is normal return address; if zero, tape specifies return

1.2.1.2 Compiler

- a) Provisions were made to incorporate programmer macros into the Mercury SOS as system macros (see subsection 1.2.3). Core storage was made available for these system macros by removing certain instructions not used in Mercury (the instructions referring to data channels E, F, G, and H, the magnetic drum, and the cathode ray tube). The Mercury SOS tape does not presently include the Mercury programmer macros.
- b) SCAT was changed to recognize the machine instruction PSLF (present sense lines, channel F), which activates the subchannels of the Data Communications Channel (DCC).
- c) The maximum dictionary size was reduced from 8192 entries in the Share system to 8000 for Mercury.
- d) The following 65K instructions have been added: TIA, TIB, SEA, SEB, IFT, and EFT.
- e) 7094 instructions have been added.

- f) The highest location from the previous job is passed on to the following job. The first job must be preceded by a FIELD card. The highest location may be picked up in the following job by placing a BSS card after a TCD card with no intervening ORG card. The BSS should have a location symbol for reference purposes.
- g) All 6-letter symbols beginning with SYS are passed on to successive jobs when the first job has been preceded by a FIELD card. A maximum of 600 such symbols may be passed on. If a passed-on symbol is used as a location in a job, the location symbol overrides the passed-on symbol for that job only.
- h) ORG, BSS, BES, EQU, and SYN can select the highest value of several choices. For example, ORG A=B=C. If B is the highest value, the ORG occurs at B. Address fields may be complex, such as A+B=C/D=D+F. (With these instructions, the equal sign is part of the format and does not denote equality.)

1.2.1.3 Modify and Load

- a) The PSLF instruction was added to the decode and list files (see \underline{b} under "Compiler."
- b) Deletions similar to those in Compiler were made from SOS to reserve storage for additional Mercury programmer macros.
- c) The decode and lister files were also altered to compile squoze cards using SQZ or SQZ RB. SQZ and LBR have a better chance of succeeding if they occur near the start of the program.
- d) The Modify and Load supervisory controller and symbol assigner files were altered to use certain symbols internal to SOS. These are mentioned under 1.2.1.1, items c and d.
- e) LBR cards may be altered out of a program at Modify and Load time; however, there is presently no way to alter in a routine from the Library tape.
- f) The maximum number of TCD cards acceptable has been raised from 50 to 500.
- g) A card of the form A BSS A is now acceptable.
- h) Modifications were made which enable SOS to assign origins other than SYSORG to program files. A TCD followed by a BSS 1 (with no intervening symbols or ORG cards) will cause the origin of the BSS to

be assigned from a table starting at location SYSFLO in Monitor. The maximum number of these origins now available is 28. Additional ones would be assigned from SYSORG. The table must be initialized by one job with desired origins to be assigned in succeeding jobs. The table would appear as:

PZE A,,B PZE C,,D PZE E

if five starting locations were desired by this method. This feature is primarily applicable to the dual-compilation scheme used for the Mercury Programming System. However, the operator should recall SOS from tape after using this feature.

- i) The following 65K instructions have been added: TIA, TIB, SEA, SEB, IFT, and EFT.
- j) 7094 instructions have been added.
- k) Highest location pass-on, symbol pass-on, and highest value selected in principal pseudo-ops have been incorporated. See subsection 1.2.1.2 f), g), and h) for a discussion of these features.)
- 1) If a symbol occurs after a wrap around, the new SOS tape will announce "Core Wrap Around" once for each instance of wrap around.

1.2.1.4 Debug

- a) The following 65K instructions have been added: TIA, TIB, SEA, SEB, IFT, and EFT.
- b) 7094 instructions have been added.
- c) A FIELD card prevents the SNAP routine from being loaded, therefore, the programmer may not use CORE, PANEL, etc., unless he makes provisions to load SNAP.
- d) Snaps may be taken in an intermixed fashion from different jobs. Preedited dictionaries must be on A10 and the first pre-edited dictionary must be loaded before calling SNAPTRAN. B2 tagging records identify which dictionary is to be used. The tagging record consists of one word in the following format: PZE JOB#,,21.
- e) If the program has left multiple-tag modes, the panels will be all of seven index registers.

1.2.1.5 Input/Output

There are no deviations from the Share Input/Output section.

1.2.2 MERCURY SOS TAPE FEATURES

The following paragraphs present a partial listing of the features of the Mercury SOS tape:

- a) The JOB card is reproduced at the start of all decks punched during CPL, PS and PA runs. If the no-punch option is exercised, the JOB card is represented on tape as one binary record. (A 1 appears in column 1 to eject paper on the on-line printer for each new job.) The JOB card should be removed before sequencing the deck but may be used as the first card in the deck setup for future execution runs.
- b) LBR cards need not appear in the symbolic deck in the same order as the routines appear on the Library tape.
- c) The date may be inserted into a CPL or PS listing using the control card, DATE, and the month/day/year in columns 16-21, as "bbbbbbb DATE bbbb 121560." The date can also be entered manually on the console keys in the same manner; however, the DATE card overrules the keys. The DATE card should immediately follow the JOB card.
- d) The "rescue" operation saves any debug macro output which may have been written on B2 before a program came to an unexpected stop or loop. The rescue operation procedures are as follows:
 - 1) Rewind A1
 - 2) Depress sense switch #4
 - 3) Press LOAD TAPE

A memory dump from 3084_{10} to 32767_{10} is taken first. (To skip the dump, press CLEAR before pressing LOAD TAPE.) TRYING TO RECOVER SNAPS AFTER PROGRAM SCRAMBLED MEMORY is printed on-line.

e) Column binary input and output (squoze or absolute) cards are the normal modes of operation and therefore may be processed on-line. Row binary squoze cards must be read on-line, and the appropriate control card must have RB in columns 16 and 17. The control card, CPLRB, must be used for row binary squoze output from a compilation. For row binary (squoze or absolute) output from an execution,

- RB must be in columns 18 and 19 of the appropriate control card (LG, PA, PS or LS).
- f) A CORE macro consolidates large groups of consecutive identical words; the message, WORDS OFOMITTED, appears instead of the individual locations (formerly this was true only for consecutive zeroes). Symbols attached to the omitted locations are not used.
- g) The EJECT macro is inoperative at the top of a page unless two or more EJECT's occur in succession.
- h) Symbolic cards can be used to provide comments both on-and off-line if columns 8-13 are blank and the cards are inserted between the JOB card and the next control card. By this means, a message can be given to the operator without using machine instructions.
- i) One computer run may combine a CPL and GO, or PS and GO, or LS and GO job. The deck setup for this is:
 - 1) JOB card
 - 2) DATE and comments cards (both optional)
 - 3) CPL, PS or LS control card
 - 4) Symbolic or squoze deck
 - 5) Blank card
 - 6) GO card
 - 7) PAUSE card
- j) The variable fields of DETAIL and SPACE are interpreted.
- k) A tape bad-spot routine has been added to all sections of SOS and should reduce tape failures.
- 1) A squoze deck to be listed using LS may now contain mods (except column binary on-line). The new date appears on the listing, and the job is treated as PS.
- m) A PS run with a squoze deck without mods may be made. The MOD and ENDMOD cards may be omitted, but an error message is printed and the job is treated as LS.
- n) Two or more tapes may be used as input to the compiler for one job.
 This splitting is accomplished by placing an ENDTAP card on each

incomplete tape except the last. When encountered by the compiler, the ENDTAP card causes the following message to be printed on-line: END OF SYSPIT. MOUNT THE NEXT SYSPIT AND PRESS START.

o) Commentary at the start of the program can now be deleted by using ERASE. ERASE (MACRO) only removes the macro skeleton from a punched squoze deck. It has no effect on the macro expansions in the program.

1.2.3 MERCURY SOS LIMITATIONS

The following is a list of the Mercury SOS Systems' limitations.

- a) A program should not origin below 3000_{10} and cannot origin at zero.
- b) The following insertions are inoperative at Modify and Load time: DUP, LBR, SQZ, EXEMPT, Extended Range Instructions programmer macros unless they are redefined, and ETC after a VFD. HEAD may be altered in and out on non-PS runs (as well as PS runs), except column binary on-line. This requires an A5 and makes a listing since it causes a PS operation. All 7094 instructions may be inserted except indirectly addressed I/O commands. (For example, IORP* A,, B would have to be altered in as IORP A,4, B. The resulting SOS error indication should be ignored.)
- c) SYN, EQU, or BOOL is omitted if the location symbol or variable field is missing.
- d) Principal pseudo-operations* may not be modified with, nor included in, the range of a CHANGE.
- e) If a BSS or BES pseudo-operation has both an associated symbol and a debugging format code, the latter is attached to the symbol associated with the BSS or BES. If either a BSS or BES has no associated symbol, the debugging format code, if present, is ignored. If a BSS or BES has an associated symbol but no debugging format code, the code is attached to the symbol, becomes the prevailing code for compilation and will be octal for Modify and Load.
- f) Depressing sense switch 3 causes the Modify and Load and Compiler errors to be printed on-line. However, during GO time, portions of the debug macro output may be selected for on-line printing by toggling sense switch 3.
- g) Any insertion which is attempted in the middle or at the end of a block of remarks cards or listing pseudo-operations is always made before the remarks or pseudo-operations.

- h) If an MSE or PSE with a blank variable field is inserted, it is listed incorrectly.
- i) If an illegal subfield exists in the variable field of a DEC, a zero is inserted for that subfield and an error message is printed, but subsequent fields may be dropped or improperly converted.
- j) Principal pseudo-operations* cannot occur within the range of a DUP.
- k) There is a limit to the number of modifications which may be made for an execution run. The number of permissible modifications is a function of the core size, length of program, the type of modification, and the number and length of insertions; where "mods" is the sum of ALTER's, CHANGE's, ERASE's, ASSIGN's, dictionary, introduction, and footnote entries. Thus, there is no practical reason for determining the numerical value of this limit. If the phrase MODIFICATIONS EXCEED LIMITS appears after the ENDMOD card, the number of changes should be condensed or a recompilation made. The SUMARY program is provided to determine the nearness of the modification limit, and to advise on the recompilation schedule.
- 1) A VFD with its ETC's may not specify more than 66 subfields or generate more than 50 words.
- m) Squoze decks (tapes) prepared from CPL or PS runs using a current SOS tape cannot be executed using an earlier version of the SOS tape.
- n) The limitations of programmer macros are:
 - 1) The name of a macro may not contain any numeric characters.
 - 2) The macro definition does not appear in the Compiler listing; however, using a DETAIL prior to calling the macro causes all the generated instructions to be listed.
 - 3) The macros are not available at Modify and Load time. To insert a macro, it must be redefined in the modification packet immediately following the MOD card and before any ALTER's or CHANGE's.
 - 4) The elements in the variable field of macros appearing within programmer macros are prefixed by a plus sign.

^{*}BES, BOOL, BSS, END, EQU, HEAD, ORG, SYN, and TCD are the principal pseudo-operations.

- 5) ERASE QXX (where QXX is a macro name) is not operative.
- 6) No programmer macro may contain an operation of the form A\$B (where A is a parameter of the macro).
- 7) The maximum number of programmer macros which can be defined by Modify and Load is 20.
- 8) A single macro definition may not employ more than 32 parameters.

1.2.4 INCORPORATION OF MERCURY PROGRAMMER MACROS INTO SOS AS SYSTEM MACROS

The Mercury SOS System tape does not include any of the programmer macros of the Mercury Programming System. The definition of each macro must, therefore, be inserted as modifications during each execution run if any CHANGE or ALTER inserts a use of that macro. However, it is possible to incorporate these macros into SOS so they can be available (as are CORE and PANEL) without redefinition at Modify and Load time.

SOS macros may be defined either by skeletons* or generators.** The definition of a macro by the skeleton is preferable to definition by a generator, since the former requires less coding and is the method used to define programmer macros. A generator would have to be used for macros which do not adhere to the restrictions placed on programmer macros (for example, a macro such as BEGIN, which does not always generate the same number of instructions). The discussion which follows is confined to macros defined by skeletons. Additional information is available through SHARE.

^{*}The "skeleton" is a series of octal data words which describes the generated instructions according to a formula.

^{**}A macro "generator" is a program added to the C1 and MO files of SOS to produce, in BCD, the instructions of a macro. When the generator receives control, the identity of the macro has been determined and the starting column and number of characters in the parameters have been tabulated. After producing each instruction, the generator relinquishes control to permit encoding of the instruction as if it were symbolic input. When control returns after the processing of the last generated instruction, the generator exits by a transfer to the basic routine to resume processing symbolic input.

The first step in incorporating a programmer macro is to compile the symbolic definition of the macro. The deck for this computer pass might consist of the following cards:

	JOB	
	CPLRB	
QXSUM	MACRO	A, B, C
	CLA	A
В	ADD	3004
	C	3010
	END	
START	QXSUM	D, E, STO
	TRA	SYSERR
D	DEC	1
	END	START
	blank card	
	PAUSE	

The resulting row binary squoze deck includes a macro name table as the card preceding the blank card, and the macro skeleton table as the card(s) immediately following the blank card within the squoze deck. For each macro that is compiled (one, in the above example), the macro name table contains two adjacent words of information:

Word 1-NAME00: the macro name, left-justified, to provide six characters.

Word 2—Address: relative position of the first word of the skeleton in the skeleton table (zero for the first skeleton).

Word 2-Decrement: number of words in the skeleton.

For each macro to be inserted, three modifications to the SCAT1 or C1 file of the SOS Compiler and three modifications to the MLMOD1 or MO file of SOS Modify and Load are required:

- a) BCI 1, NAME00, where NAME consists of one to six alphabetical characters, should be inserted in the tables whose origins are at OPTBL in C1 and TOPCO + 1 in MO.
- b) An information word, whose structure is given below, should be inserted in the tables which origin at FLAG in C1 and TOPAN + 1 in MO.

	C1 Information Word
$\underline{\mathrm{Bits}}$	<u>Use</u>
S, 5-11, 20	Available to the macro generator
1, 3, 4	0
2, 19	1
12-17	Minimum number of parameters which may be specified
18	0—macro defined by skeleton
_•	1—macro defined by generator
21-35	Location of skeleton or generator
22 00	Location of sketeton of generator
]	MO Information Word
	MO Intol mation wold
Bits	<u>Use</u>
Bits S-5	
S-5	<u>Use</u> 011 001
	Use
S-5 6-15, 17, 20	Use 011 001 Available to the macro generator
S-5 6-15, 17, 20 16 19	Use 011 001 Available to the macro generator 0 1
S-5 6-15, 17, 20 16	Use 011 001 Available to the macro generator

01 Information 337 and

c) Octal data words (using the pseudo-operation, OCT, and the data from the skeleton table in the squoze deck obtained above) for the macro skeleton should be inserted after the remark MACRO SKELETONS FOR SYSTEM MACROS in C1 and MO (at alter numbers 8543₈ and 7321₈, respectively). The first octal word should be given the location symbol specified in bits 21-35 of the information word described in b) above.

(NOTE: Reference to listings without modifications of the C1 and MO files should be made to obtain the alter numbers for the above insertions.)

With modifications, the C1 and MO squoze decks must be punched absolute to obtain absolute (row) binary decks using a control card with the following format:

Column	Punch	
8-9	PA	
18-19	RB	

A punch squoze run is necessary to obtain a listing with modifications. The new absolute binary decks are used to update the SOS System tape using the IBWR2 program and the appropriate Hollerith cards preceding each binary deck.

1.2.5 COMPONENTS OF THE MERCURY PROGRAMMING SYSTEM

The complete Mercury SOS Programming System consists of:

- a) The Mercury SOS System tape which incorporates modifications from New York through SHARE Distribution Number 32 and various other changes. (Individual files on this tape are listed in Table 1-1. Other features of the tape are discussed in subsection 1.2.2.)
- b) The Mercury Library and Regular Library tapes.
- c) The New York SOS System tape which includes all sections exactly as they have come from SHARE (through Distribution Number 32).
- d) Two decks, each consisting of approximately 3000 absolute row binary cards. One deck is used to write the Mercury System tape and the other is used for the SHARE System tape.
- e) Each system (Mercury and SHARE) in column binary squoze cards with appropriate modifications. Each deck contains approximately 10,000 squoze cards.
- f) A folder of listings for the sections to the SOS System.

1.2.6 SHARE SYSTEM TAPE WRITER PROGRAM (IBWR1)

IBWR1 is a self-loading program that accepts the SOS master cards in IBM 709/7094 binary format and produces an SOS System tape (A1).

1.2.6.1 Input Requirements

Input to IBWR1 consists of symbolic control cards and the binary programs which comprise the SOS System. The cards are described in the order in which they must be supplied as input:

- a) Sequence Cards (SEQ punched in columns 8-10)—contain the system file identification in columns 16-21. The order of these sequence cards specifies the order in which the corresponding system files are to be written on the SOS System tape—these cards must be arranged in the exact order intended for the files on the tape, and SOS Monitor (file identification, MON) must precede all other files on the SOS System tape.
- b) Program Deck (name card and program)—each name card is a preface for, and must precede, the program it represents. The system file identification (from columns 16-21 of the sequence card) is punched in

TABLE 1-1. INDIVIDUAL FILES, MERCURY SOS SYSTEM TAPE

File No.	File Name	Sequence Name	Section	Inclusive Decimal Locations in Core Storage	Number of Locations Used	Number of Records
1			Tape Loader			1
2	MN	MON	Monitor	32767-1, 8, 10-1796	1791	12
3	M1	MLSUPR	Modify & Load	28672-29573	902	19
4	МО	MLMOD1		22978-31678	8701	35
5	МЗ	MLMOD2		27617-31546	3930	15
6	M7	MLPCHI		29937-31616 31667-32182	2196	10
7	M7	MLPCH2		28672-31401	2731	10
8	M7	MLPCH3		31117-31626	510	4
9	M7	MLPCH4		30067-31 <i>5</i> 71	1505	-5
10	M7	MLPCH5		30998-32198	1211	6
11	M4	MLASGN	!	28672-29790	1119	6
12	M5	MLDCOD		28160-31564	3405	13
13	M6	MLDERP		28672-29740	1069	9
14	M8	MLLIST		5400-10800	5401	45
15	М9	MLEROR		28672-29526	855	7
16	DI	SNP1	Debug	1798-2961	1164	8
17	D3	DDE		2000-2408 32618-32688	480	8
18	D2	SNP2		2000-5992	3993	19
19	С1	SCAT1	Compiler	2000-16752 24770-25295	15279 + BSS	45
20	C2	SCAT2		2000-11998	9999+C1 BSS	13
21	DA	DS1	Input/Output	9000-10449	1450	13
22	IN	INTRAN		3000-8147	5148	25
23	ОТ	OUTRAN	1	10500-14505	4006	18
24	TM	TM		14500-15108	609	6
25	į		EOT File			1

columns 9-13 of the name card. The program follows the name card and is in absolute row binary format. Each name card, and corresponding program, represents one file in the SOS System tape. The program decks must be in order as designated by their related sequence cards.

c) Blank Card—indicates to IBWR1 the termination of input. No input cards may follow the blank card.

1.2.6.2 Output Requirements

Output from IBWR1 is the complete SOS System tape. (The composition, by files, of the Mercury SOS System tape is illustrated in Table 1-1.)

1.2.6.3 Usage

- a) Operator's Procedures:
 - 1) Ready the on-line card reader with the IBWR1 program followed by the SOS system input cards.
 - 2) Ready a blank tape on A1.
 - 3) Ready the on-line printer with the SHARE #2 board.
 - 4) Press LOAD CARDS to load the IBWR1 program. IBWR1 reads in the SOS System input cards and writes the SOS System tape on A1.
 - 5) The program halts at 00572_8 .
 - 6) Press START to check the SOS System tape. The tape is reread and checked for redundancy and valid checksum recording. An on-line printout of file identifications and error indications, if any, is furnished. Other on-line printouts include the total number of tape files written—excluding the bootstrap loader and the end-of-tape files—the number of tape records, and any read/write checks encountered.
 - 7) The SOS System tape, A1, is rewound and the program comes to a final halt at 01033_o.
- b) Error Conditions—a complete list of program error stops is presented in Table 1-2.

TABLE 1-2. IBWR1 PROGRAM STOPS

TABLE 1-2. IDWKI PROGRAM STOPS		
Octal Location	Meaning and Procedure	
00201	Illegal system file name punched into sequence card. Correct and start again.	
00212	No sequence cards in system deck. Furnish same as required and start again.	
00433	System deck binary program card contains error in checksum. If SS1 is up, press START button to cause card-punched sum to be ignored (this does not affect validity of subsequent checksum recordings on System tape if the card in question is correctly punched in all other respects). If SS1 is down and SS2 is up, reload card in question and press START to reread and recheck. If SS2 is down, reload entire subdeck (including related name card) containing card in question to regenerate correct System tape file.	
00522	System deck binary program card punched with incorrect word count (i.e., exceeding 27 ₈). See 00433 stop regarding SS2 setting.	
00533	System sequence cards and name cards in asynchronous order. Place cards in order and restart.	
00546	Illegal system name punched in name card. Correct and start again.	
00563	Sequence card present for which no corresponding program exists in system deck. Eliminate sequence card or furnish program and restart.	
00572	System tape has just been written and rewound. Press console START button to initiate full System tape checking.	
00612	Bad tape logic encountered during course of file recount procedure. Restart. If this stop recurs, select different physical tape unit for A1.	
00660	Five successive failures occurred in attempting to write a System tape record. If SS1 is up, press START to write record as is. If SS1 is down, press START to attempt rereading of record five additional times.	
00675	Bad tape logic encountered in attempting to pass over EOF mark following bootstrap file. See 00612 stop.	
01004	Five successive failures in attempting to read System tape record. If SS1 is up, press START to ignore error. If SS1 is down, press START to reread record.	
. 01033	Final IBWR1 stop. System tape is on tape unit A1.	
01041	Bad tape logic—EOF does not occur immediately following TCD (transfer card) record. Restart from the beginning (possibly after having switched A1 to a different tape unit).	

1.2.7 SHARE SYSTEM TAPE EDITOR PROGRAM (IBWR2)

After the SOS System tape has been written by IBWR1, any or all of the files of this tape may be edited and replaced by using the self-loading editing program, IBWR2. This program may also be used to duplicate the SOS System Tape.

1.2.7.1 Input Requirements

Input to IBWR2 must be in the same format and have the same relative order as the input to IBWR1: sequence cards, program decks (name card and program), and a blank card as the last card. Each program represented by a sequence card and a program deck replaces the program of the same name on the SOS System tape.

The SOS System tape to be updated must be placed on tape unit B1.

If IBWR2 is used to duplicate the SOS System tape, the only input other than the SOS System tape on B1 consists of a blank card. This card is placed immediately after the IBWR2 self-loading program deck.

1.2.7.2 Output Requirements

Output from IBWR2 is an updated (or duplicate) SOS System tape. The order of files on the updated tape will be identical to the order on the input tape.

1.2.7.3 Usage

- a) Operator's Procedures:
 - 1) Ready the old SOS System tape on B1.
 - 2) Ready a blank tape for output on A1.
 - 3) Ready the IBWR2 program in the card reader.
 - 4) Ready the input updated programs with at least one blank card after the last program. Each updated program consists of its appropriate name card, binary program cards, and associated transfer card:
 - (a) If input is to be on-line, place the updated (row binary) cards in the card reader behind IBWR2.

- (b) If input is to be from tape, go card-to-tape with the updated (columnar binary) cards. Place this tape on A3 and depress sense switch 2.
- 5) Ready the on-line printer with either the Share #2 board or the MOCKDONALD printer board.
- 6) Press LOAD CARDS.
- 7) The program comes to a final halt at 01430 with the updated (or duplicated SOS System) tape rewound on A1.
- b) Error Conditions—a complete listing of program error stops is presented in Table 1-3.

TABLE 1-3. IBWR2 PROGRAM STOPS

	T
Octal Location	Meaning and Procedure
01045	Illegal system name punched in name card. Correct and start again.
01053	Program indicated on name card is not on old SOS System tape. IBWR2 cannot update program which does not exist on SOS System tape.
01064	A file identification number on the old System tape exceeds 32, 767 ₁₀ . Start again. If stop recurs, switch B1 to another tape unit and try again. If still unsuccessful, the old System tape is probably no longer usable. A new System tape should be written with IBWR1.
01067	File to be updated has been bypassed on old System tape. Check to ensure that order of updated program decks corresponds to file order on old SOS System tape.
01244	709 binary card punched with incorrect checksum. Correct and start again.
01247	704 binary card punched with incorrect checksum. Correct and start again.
01324	704 binary card punched with bad word count, i.e., exceeding 26g. Correct and start again.
01325	709 binary card punched with bad word count, i.e., exceeding 278. Correct and start again.
01430	Final IBWR2 stop. Updated SOS System tape rewound on A1.
01453	Read check on B1. If SS6 is up, press START button to attempt to reread B1; if SS6 is down, press START to continue, ignoring error.
01461	Write check on A1. If SS6 is up, press START button to rewrite and recheck; if SS6 is down, press START to continue, ignoring error.
01471	See stop location 01461
01477	See stop location 01461
01 5 07	See stop location 01453
01515	See stop location 01461
01523	See stop location 01453
01532	See stop location 01453
01540	See stop location 01461
01546	See stop location 01453
01554	See stop location 01453
01562	See stop location 01453
01602	See stop location 01453
01616	Redundancy on A3 (SYSPIT). Start again. Persistent redundancy indicates blank card does not follow input programs. Rewrite SYSPIT before repeating.

1.2.8 CONSOLE OPERATION DATA FOR MERCURY SOS

Tape Unit Assignments; Equipment System Symbols

<u>Unit</u>		System Symbol
A1	System Tape	SYSTAP
$\mathbf{A2}$	Peripheral Output Tape	SYSPOT
A 3	Peripheral Input Tape	SYSPIT
A4	Library Tape	SYSLBR
A 5	Peripheral Punch	SYSPPT
B1	Mediary Input	SYSMIT
B2	Mediary Output	SYSMOT
Reader	• -	SYSCRD
Printer		SYSPRT
Punch		SYSPCH

Sense Switch Settings

SS1	Up Down	Tape input Card input
SS2		Not used
SS3	Up Down	Print only control cards and error state- ments on-line On-line printing of everything
SS4	Up Down	Compile or Execution run "Rescue" operation to take core dump and capture SNAPS from B2
SS5	Up Down	Normal mode Suppress SYSPOT output during CPL
SS6	Up Down	Punch off-line, columnar binary Punch on-line, columnar or row binary, as specified in control cards

Program Stops

The following is a complete list of Mercury SOS program stops:

Octal Locations	Meaning		
00173	PAUSE card.		
01644	End-of-file (EOF) on tape unit A3 or on-line card reader.		
01746	End-of-tape (EOT) on tape unit A2.		
02420	End-of-tape (EOT) on tape unit A5.		
22422	Tape unit A2 is full and rewinding; set new tape on A2 and press START.		
77202	Redundancy check in reading or writing during modifications punch squoze, or decoding. To determine unit failing:		
	1) Display the sense indicators.		
	2) Subtract the decrement of the indicator contents from 77273 ₈ .		
	3) Display the resulting location.		
	4) The contents of the address of the resulting location is the octal code for the unit failing (1201 is for A1, etc.).		
	5) Push START to accept the last attempt and continue.		

1.3 SOS LIBRARY TAPE

The SOS Library tape consists of utility computational subroutines which are, in effect, programming aids to reduce the amount of programmer coding needed to include a specific mathematical process in a program.

The programs included on the library tape are listed in Table 1-4. The number of locations and the time required by each program are also presented in the table. LBRWR, the program used to generate a library tape, is described at the end of this section.

To allow various programs of the Mercury Programming System to share some library programs (and thus conserve core storage), it was necessary to recode these library programs to protect intermediate or temporary results from program interrupts and subsequent entry into the library program before the interrupted pass could be completed. This recoding requires two library tapes:

- a) "Regular SOS Library"—a tape on which each program contains its own temporary storage. This tape is used with SOS in compiling programs for testing (whether or not the programs are later incorporated into the Mercury Programming System).
- "Mercury SOS Library"—a tape on which the programs lack individual temporary storage. A program compiled using this tape (such as the Mercury Programming System) must provide such storage by including a COMMON BSS 5, O and arranging to preserve the five COMMON locations any time a program interrupt (trap) occurs. LBR cards of the form LBR U1SICO (with a blank location field) must be inserted in an unheaded section of the program, though not necessarily in the order in which the programs appear on the Library tape. Within any headed sections using library programs, cards of the form SIN SYN \$SIN must be included for each exempt symbol. One or more "exempt symbols" are listed for each library subroutine. A card of the form LBR U1SICO will cause all instructions in the subroutine to be relativised to those symbols specifically exempted from relativization. (To see the expanded routine in this form, a DETAIL card should precede the LBR card. A LIST card following the LBR will restore the remainder of the listing to the undetailed mode. To suppress relativization and thereby bring in all symbols from the original symbolic version of the subroutine, a card of the form LBR U1SICO, U should be used.)

The exempted symbols are normally those to which the user must have access in the variable field of his TSX instruction. The indication of which symbols are to be exempted is made at the time the library tape is prepared.

TABLE 1-4. MERCURY SOS LIBRARY TAPE PROGRAMS

			· · · · · · · · · · · · · · · · · · ·		
Name	Description	Address of TSX Instruction	Storage Required (Decimal)	Average Time Required (milliseconds)	
				IBM 709	IBM 7094
U1SICO	Sine/Cosine	SIN or COS	99	2.20	0.260
UIEXPE	Exponential	EXP	52*	3.05	0.200
UISQRT	Square Root	SQRT	43*	1.15	0.150
UILOGE	Natural Logarithm	LN	419	0.85	0.09
UIATAB	Arctangent A/B (requires U1ATNA)**	ATNAB	31*	3.01	0.080
UIATNA	Arctangent A	ATNA	81*	2.44	0.230
UIASCO	Arcsine/Arccosine	ARSIN or ARCOS	117*	1.75	0.251
UITACO	Tangent/Cotangent	TAN or COT	88*	2.30	0.24
UIFXPT	Fix a Floating-Point Number	FIX	25*	0.42	0.056
UIFLPT	Float a Fixed-Point Number	FLOAT	21*	0.65	0.071
U3DOTP	Vector Dot Product	U3DOT	18*	0.98	0.102
U3XPRO	Vector Cross Product	U3XPR	33*	1.83	0.199
U3MATM	Matrix Multiplication	U3MAT	81	2.80	0.31
UA1LSC	Converts XYZ Coordinates to RAE (requires UISQRT, UIATAB, UISICO; each used twice)**	A1LSC	209	49.4	4.9
U7INTP	Six-Point Langrangian Interpolation	UINTP	258*	26.37	4.02
U3VMAG	Vector Magnitude (requires U1SQRT)**	VMAG	19*	2.46	0.130
U3VPRO	Vector Triple Cross Product	VPRO	80	4.41	0.49
	Unit Vector (requires UISQRT)**	UNITV	30*	3.52	0.374
U7RVTH	Elliptic Motion Computation (requires UISQRT, UIATAB, U3DOTP, U3VMAG, U7ASKE)**	C9RVTH or C9RVT2	380	18.46	2.91
U7ASKE	Solve Kepler's Equation (requires U1SICO)**	C9ASK E	104	18.68	3.09
UAMSCP	Sub Spacecraft Position (requires UISQRT, UIATAB)***	A3MSCP or A3MSCP+12	119	8.71	1.35

^{*}Recoded for the Mercury SOS Library tape to store temporary results in a 5-cell table common to all indicated programs. The temporary storage cells are not included in the storage listing above.

^{**}When additional library programs are necessary for any library program, the "Average Time Required" specifies the total time of execution, i.e., the time of the main library program plus the time of the other library programs used by it as subroutines. If equivalent programs are substituted, the time of the main library program alone may be obtained by subtracting the time required by the subroutines from the time listed for the main program.

Caution: If a location symbol is assigned to the LBR card itself, it will replace the symbol attached to the first instruction generated by the library routine rather than be made synonymous with it.

1.3.1 SINE/COSINE SUBROUTINE (U1SICO)

U1SICO computes the sine or cosine of an angle.

1.3.1.1 Input Requirements

The normalized floating-point angle in radians must be in the accumulator upon entry into U1SICO. The absolute value of the argument must be less than 236.

1.3.1.2 Output Requirements

The normalized floating-point sine or cosine is present in the accumulator upon exit from U1SICO. The execution of U1SICO turns on the AC overflow indicator.

1.3.1.3 Method

This subroutine was adapted from the SHARE-distributed subroutine IBSIN1. Sine x is evaluated from one of two continued fractions.

a) For $|x| \leq 3$:

Sine x = x
$$\left(19.8459242619 + \frac{x^2}{2} + \frac{1042.92670814}{x^2 + 50.0302454854}\right)$$

b) For $.3 < |x| \le 1.3$:

Sine
$$x = 19.47714945237 - 2m^2$$

$$-\frac{3276.33995164 - 320m^{2}}{m^{2} + 82.5803019956 + \frac{2287.44319569}{m^{2} + 24.1448946943}}$$

where
$$m = (\pi/2 - x)$$

Cosine x is evaluated as sine $(\pi/2 - x)$

1.3.1.4 Usage

a) Calling Sequence:

Location	Operation	Address, Tag, Decrement
alpha + 1	TSX	SIN, 4 or COS, 4 Error return
+ 2		Normal return

- b) Error Conditions—an error return is made for any absolute value of the argument equal to or greater than 2³⁶, or if the divide check indicator has been turned on during the execution of the program.
- c) Storage Required—99 cells (plus five cells of common storage).
- d) Time Required:

Sine 0.22 milliseconds Cosine 0.260 milliseconds

- e) Accuracy—the relative error is less than $1/2 \times 10^{-8}$.
- f) Exempt Symbols—SIN, COS.
- g) Library Identification—LBR U1SICO.

1.3.2 EXPONENTIAL SUBROUTINE (U1EXPE)

U1EXPE computes the value of the natural exponential function e^X.

1.3.2.1 Input Requirements

The normalized single-precision floating-point argument (x) must be in the accumulator upon entry into U1EXPE. The absolute value of x cannot exceed 88.088.

1.3.2.2 Output Requirements

The normalized floating-point product is present in the accumulator upon exit from U1EXPE.

1.3.2.3 Method

This subroutine uses a rational approximation developed by E. G. Kegbetliantz (IBM Journal of Research and Development, April 1957, pp. 110-115), and was adopted from the SHARE-distributed subroutine IBFXP.

From the Pade' table for e^{x} , the diagonal elements Pmm (x) = Pm (x) are chosen so $e^{x} = \frac{Pm(x)}{Pm(-x)} + e^{x} Rm(x)$

(2m) ! Pm (x) = m !
$$\sum_{s=0}^{m} \frac{(2m-s)! x^{s}}{(s! (m-s)!)}$$

and
$$e^{x} = 2^{x \log_2 e} = 2^{m + f}$$

where $0 \le f < 1$.

Therefore, $e^{X} = 2^{m}2^{f}$

where m is the binary characteristic of e^X.

The value of 2^f is computed for the interval $0 \le f < 1$.

By choosing m=4 in the rational approximation, the bound for the relative error is Rm (x) $< 3 \cdot 10^{-9}$.

Therefore,

$$2f = 1 + 2f \left[A + Bf^2 - f C (f^2 + D)^{-1}\right]^{-1}$$
.

The values for the constants are:

A = 9.95459578 B = 0.03465735903 C = 617.97226953 D = 87.417497202

 $\log_2 e = 1.44226950409$

1.3.2.4 Usage

a) Calling Sequence:

Location	Operation	Address, Tag, Decrement
alpha	TSX	EXP, 4
+ 1		Error return
+ 2		Normal return

b) Error Conditions—an error return is made with x in the accumulator for any value of x greater than 88.088. (A normal return is made with zero in the accumulator if x is less than -88.088). An error return is

also made, with e^X in the accumulator, if the divide check indicator is turned on during execution of the subroutine.

- c) Storage Required-52 cells (plus five cells of common storage).
- d) Time Required:

Average

0.200 millisecond

Maximum

0.35 millisecond

- e) Accuracy—the number of significant bits in the result is at least 155—C, where C is the characteristic of the argument x.
- f) Exempt Symbol—EXP.
- g) Library Identification—LBR U1EXPE.

1.3.3 SQUARE ROOT SUBROUTINE (U1SQRT)

U1SQRT computes the square root of a number.

1.3.3.1 Input Requirements

The accumulator must contain the normalized single-precision floating-point argument upon entry into U1SQRT.

1.3.3.2 Output Requirements

The accumulator contains the normalized floating-point square root upon exit from UISQRT. The AC overflow indicator light is turned on during the execution of the program.

1.3.3.3 Method

This subroutine was adapted from the Share-distributed subroutine IBSQ1: if the argument is zero (positive or negative), it is returned. If the argument is algebraically less than zero, an error return occurs with the argument in the accumulator.

The floating-point argument has the form $x \cdot 2^{C}$, where x is the normalized fraction part (.5 \leq x < 1) and C is the characteristic. If the characteristic is odd, it is increased by one and the fraction part is divided by two (the fraction part is shifted one bit position to the right).

Setting C = 2b,
$$\sqrt{N} \sqrt{x \cdot 2^{2b}} = 2^b \sqrt{x}$$
.

The first approximation of \sqrt{x} is found by $\sqrt{x} = Ax + B$.

For
$$0.25 \le x < .5$$
: $A = 0.875$, $B = 0.27863$

For
$$0.5 \le x < 1$$
: $A = 0.578125$. $B = 0.421875$

The first approximation is used to compute the square root (\sqrt{N}) by two Newton iterations:

$$\sqrt{N} \approx a_2 = \frac{1}{2} \left(\frac{N}{a_1} + a_1 \right)$$

where:

$$a_1 = \frac{1}{2} \left(\frac{N}{a_0} + a_0 \right)$$

$$a_0 \approx 2^b \sqrt{x}$$

1,3.3.4 Usage

a) Calling Sequence:

alpha	TSX	SQRT, 4	
+ 1		Error return	
+ 2		Normal return	

- b) Error Conditions—an error return occurs with the argument in the accumulator for all values of the argument less than negative zero. An error return also occurs if the divide check indicator light is turned on during the execution of the program. (A normal return with the argument in the accumulator is made for arguments of plus and minus zero.)
- c) Storage Required-43 cells (plus three cells of common storage).
- d) Time Required-

IBM 7094

Average Maximum 0.150 millisecond

0.118 millisecond

- e) Accuracy-26 significant bits.
- f) Exempt Symbol—SQRT.
- g) Library Identification-LBR U1SQRT.

1.3.4 NATURAL LOGARITHM SUBROUTINE (U1LOGE)

U1LOGE computes the natural logarithm of a number.

1.3.4.1 Input Requirements

The accumulator must contain the argument in floating-point format upon entry into U1LOGE. The argument must be positive and greater than zero.

1.3.4.2 Output Requirements

The accumulator contains the natural logarithm of the argument in floating-point upon exit from U1LOGE.

1.3.4.3 Method

The floating-point argument is in the form $F \cdot 2^{\mathbb{C}}$, where \mathbb{C} is the characteristic and F is the fraction part.

If
$$x = F \cdot 2^C$$
, Ln $x = Ln F + C Ln 2$, and Ln $x = Ln \frac{F}{A} + Ln A + C Ln 2$.

In the common natural logarithm series (let $\frac{\mathbf{F}}{\mathbf{A}} = \mathbf{Y}$):

Ln y =
$$2\left[\frac{y-1}{y+1} + \frac{1}{3}\left(\frac{y-1}{y+1}\right)^3 + \frac{1}{5}\left(\frac{y-1}{y+1}\right)^5 + \ldots\right]$$

If y is very nearly equal to 1, only the first term need be considered. Therefore, A is chosen (by taking the first eight bits of F and adding a 1 in the ninth bit position) very nearly equal to F, and the equation becomes:

Ln
$$x \simeq C$$
 Ln 2 + Ln A + 2 (F - A)/(F + A).

The 256 possible values of Ln A are contained in a table within U1LOGE.

1.3.4.4 Usage

a) Calling Sequence:

alpha TSX LN, 4
+ 1 Error return
+ 2 Normal return

- b) Error Conditions—an error return occurs when the argument is not greater than positive zero, and an indication is stored in the decrement of location LN: a 1 in the decrement for a negative argument, a 2 for an argument of positive zero.
- c) Storage Required-419 cells.
- d) Time Required-0.090 millisecond, average.

1.3.5 ARCTANGENT A/B SUBROUTINE (U1ATAB)

U1ATAB computes the arctangent of the quotient A divided by B.

1.3.5.1 Input Requirements

The ordinate A must be in the accumulator and the abscissa B must be in the multiplier-quotient upon entry to U1ATAB. Both numbers must be in normalized floating-point form and cannot simultaneously be equal to zero. The program must have access to U1ATNA.

1.3.5.2 Output Requirements

The accumulator contains the computed angle in floating-point radians upon exit from U1ATAB. The absolute value of the angle is equal to or less than π radians.

1.3.5.3 Method

This subroutine was adapted from the SHARE-distributed subroutine IBATN1 (modified). U1ATAB divides A by B and then uses U1ATNA to compute the absolute value of the quotient obtained. U1ATAB places the angle in the proper quadrant.

1.3.5.4 Usage

a) Calling Sequence:

alpha	TSX	ATNAB, 4
+ 1		Error return
+ 2		Normal return

- b) Error Conditions—an error return occurs if A = B = 0.
- c) Storage Required—31 cells (plus five cells of common storage), not including the subroutine U1ATNA.
- d) Time Required (includes execution of U1ATNA)-0.080 millisecond, average.
- e) Accuracy—26 significant bits.
- f) Exempt Symbol—ATNAB.
- g) Library Identification-LBR U1ATAB.

1.3.6 ARCTANGENT SUBROUTINE (U1ATNA)

U1ATNA computes the arctangent of a number.

1.3.6.1 Input Requirements

The accumulator must contain the argument in floating-point format upon entry into U1ATNA.

1.3.6.2 Output Requirements

The accumulator contains the computed angle in floating-point radians upon exit from U1ATNA. The absolute value of the angle is equal to or less than $\pi/2$ radians.

1.3.6.3 Method

U1ATNA uses a rational approximation method developed by E. G. Kogbet-liantz (IBM Journal of Research and Development, January 1958, pp. 43-53), and was adapted from the Share-distributed subroutine IBATN1.

Given: $\theta = \arctan x$. If the absolute value of x is greater than 2^{27} , θ is assumed to be $\pi/2$. If the absolute value of x is less than 2^{-13} , θ is assumed to be equal to x.

If neither of the above conditions is satisfied, x is divided into five subintervals and the arctangent is computed from the following formulas:

$$\theta = \frac{T_{X}}{\frac{-B}{T_{X}^{2} + A} + T_{X}^{2} + C} + K_{X}$$

where:

A = 0.051119459

B = 0.0027099425

C = 0.21664913599

 $T_{\mathbf{x}}$ may have two forms, depending upon the range of x.

For $|\mathbf{x}| < 0.1763298071$:

$$T_{X} = 0.16363636363X$$

For $|x| \ge 0.1763298071$:

$$T_{X} = \frac{-M_{X}}{x + N_{X}} + L_{X}$$

The values of the constant terms for each range are given below.

a) For |x| < 0.1763298071 ($|\theta| < 10^{\circ}$):

$$K_{\mathbf{x}} = 0$$

b) For $0.1763298071 \le |x| < 0.5530526919$ ($10^{\circ} < |\theta| < 30^{\circ}$):

 $L_{x} = 0.44958721409$

 $M_{x} = 1.398867082$

 $N_{\downarrow} = 2.7474774195$

 $K_{x} = 0.3490658504$

c) For $0.5530526919 \le |x| < 1.1917535926$ ($30^{\circ} \le |\theta| < 50^{\circ}$):

 $L_v = 0.19501422424$

 $M_{v} = 0.39604526598$

 $N_{x} = 1.1917535926$

 $K_{x} = 0.6981317008$

d) For $1.1917535926 \le |x| < 2.7474774195$ ($50^{\circ} \le |\theta| < 70^{\circ}$):

 $L_{x} = 0.094475498595$

 $M_{x} = 0.21818181818$

 $N_{v} = 0.057735026919$

 $K_{\nu} = 1.047197551$

e) For $2.7474774195 \le |x| (70^{\circ} \le |\theta|)$:

 $L_v = 0.0288535059$

 $M_{v} = 0.1687240152$

 $N_{y} = 0.17632698071$

 $K_v = 1.396263402$

1.3.6.4 Usage

a) Calling Sequence:

alpha

TSX

ATNA, 4

+ 1

Normal return

- b) Error Conditions-none.
- c) Storage Required-81 cells (plus four cells of common storage).

d) Time Required:

Average

0.230 millisecond

Maximum

0.28 millisecond

- e) Accuracy-26 significant bits.
- f) Exempt Symbol—ATNA.
- g) Library Identification-LBR U1ATNA.

1.3.7 ARCSINE/ARCCOSINE SUBROUTINE (U1ASCO)

U1ASCO computes the arcsine or the arccosine of a number.

1.3.7.1 Input Requirements

The normalized floating-point argument (whose absolute value must be ≤ 1) must be in the accumulator upon entry into U1ASCO.

1.3.7.2 Output Requirements

The floating-point answer is present in the accumulator upon exit from U1ASCO. The result is expressed in radians, within the following limits:

$$-\frac{\pi}{2} \le \arcsin \le \frac{\pi}{2}$$
; $0 \le \arccos \le \pi$.

1.3.7.3 Method

This subroutine uses a method developed by E. G. Kogbetliantz (IBM Journal of Research and Development, Vol. 2, No. 3, July 1958, pp. 218-222). This subroutine was adapted from the Share-distributed subroutines IBASN2 and IBACS2.

In the determination of arcsine x, the range of $x(0 \le |x| \le 1)$ is divided into four intervals to define $f(x) = \arcsin x$.

a) For
$$0 \le |x| \le 2^{-11}$$
: $f(x) = x$

b) For
$$2^{-11} < |x| \le \frac{\sqrt{2}}{2}$$
: $f(x) = x (A + B) \left[C + x^2 - D\left\{E - x^2\right\}^{-1}\right]^{-1}$

where:

A = 0.5249978317

B = 1.578342904

C = 3.5574340883

D = 0.3321585891

E = 1.4156902913

c) For
$$\frac{\sqrt{2}}{2} < |x| \le \frac{\sqrt{2 + \sqrt{2}}}{2}$$
: $f(x) = \frac{\pi}{4} + \frac{1}{2} p(y)$

where:

$$p(y) = y (A + B[C - y^2 - \dot{D}\{E - y^2\}^{-1}]^{-1})$$

 $y = 2x^2 - 1$

d) For
$$\frac{\sqrt{2+\sqrt{2}}}{2} < |x| < 1$$
: $f(x) = \frac{\pi}{2} - 2\sqrt{\frac{1-x}{2}} (1.085180421 - 0.0852176716x)$

The arccosine is evaluated: $\operatorname{arccosine} x = \frac{\pi}{2}$ - $\operatorname{arcsine} x$.

1.3.7.4 Usage

a) Calling Sequence:

alpha TSX ARSIN, 4 or ARCOS, 4
+ 1 Error return

+ 2 Normal return

- b) Error Conditions—an error return occurs if the absolute value of the argument exceeds one. An error return also occurs if the divide check indicator light is turned on during execution of the program.
- c) Storage Required-117 cells (plus five cells of common storage).

d) Time Required:

Range of Argument

$$0 \le |x| \le 2^{-11}$$

0.026 millisecond

$$2^{-11} < |x| \le \sqrt{\frac{2}{2}}$$

0.157 millisecond

$$\sqrt{\frac{2}{2}} < |x| \le 1$$

0.223 millisecond

(Times given apply to arcsine; computation of arccosine requires an additional 0.028 millisecond for IBM 7094.)

- e) Accuracy—the relative error is less than 6.4×10^{-7} ; in most cases the result is accurate to seven significant digits.
- f) Exempt Symbols—ARSIN, ARCOS.
- g) Library Identification-LBR U1ASCO.

1.3.8 TANGENT/COTANGENT SUBROUTINE (U1TACO)

U1TACO computes the tangent or the cotangent of an angle.

1.3.8.1 Input Requirements

The normalized floating-point argument expressed in radians must be in the accumulator upon entry into U1TACO. The argument of tangent must be within the range $|x| < 2^{35}$. The argument for the cotangent must be within the range $|x| \ge 2^{-126}$.

1.3.8.2 Output Requirements

The accumulator contains the normalized floating-point tangent or cotangent upon exit from U1TACO. Execution of the program turns on the AC overflow indicator light.

1.3.8.3 Method

The function is evaluated from a continuous fraction, based on a method developed by H. J. Maehly and E. G. Kogbetliantz. The subroutine was adapted from the Share-distributed subroutine IBTAN1.

a) The argument is reduced to the first octant, where

Cot
$$x = (3.4280166678 - 0.1015625000x^2)^{-1}$$

+ $(25.2265398966) (x^2 - 10.43274050825)^{-1} - C(x)$
For $|x| \le .15$, $C(x) = 0.526 \cdot 10^{-7}$
For $|x| > .15$, $C(x) = 0$

b) The tangent is computed from the cotangent in the first octant using the following relationships:

For
$$\frac{-\pi}{4} \le x \le \frac{\pi}{4}$$
; $\tan x = \frac{1}{\cot x}$
For $\frac{\pi}{4} \le |x| \le \frac{\pi}{2}$; $\tan x = \cot(\frac{\pi}{2} - x)$

1.3.8.4 Usage

alpha	TSX	TAN, 4 or COT, 4
+ 1		Error return
+ 2		Normal return

- b) Error Conditions—an error return occurs if the magnitude of the argument for the tangent exceeds modulo 2^{35} or if the magnitude of the argument for the cotangent is less than 2^{-126} . An error return occurs if the divide check indicator light is turned on during the execution of the program.
- c) Storage Required-88 cells (plus five cells of common storage).
- d) Time Required-0.240 millisecond, average
- e) Accuracy—the absolute error is less than $8.6 \cdot 10^{-9}$; the relative error is less than $5.26 \cdot 10^{-8}$.

- f) Exempt Symbols-TAN, COT.
- g) Library Identification-LBR U1TACO.

1.3.9 FLOATING TO FIXED-POINT CONVERSION SUBROUTINE (U1 FXPT)

U1FXPT converts a number from floating-point to fixed-point format.

1.3.9.1 Input Requirements

The accumulator must contain the floating-point number upon entry into U1FXPT. The floating-point number to be converted must satisfy the conditions: $2^{-35} \le N \le 2^{35}$.

1.3.9.2 Output Requirements

The accumulator contains the integral part of the number and the multiplier-quotient contains the fractional part of the number upon exit from U1FXPT.

1.3.9.3 Method

The floating-point number of the form $X \cdot 2^C$ is examined to determine whether the characteristic C satisfies the conditions: 93 < C < 164. If C is outside the allowable range, an error return occurs since the leading bit of the result would be lost.

The 26 bits of the fractional part of the floating-point number are shifted so the fixed-point integral is right-justified in the accumulator and the fixed-point fraction is left-justified in the multiplier-quotient. The AC and MQ are effectively treated as one register with the decimal point for the fixed-point number after accumulator bit position 35.

1.3.9.4 Usage

Location	Operation	Address, Tag, Decrement
alpha	TSX	FIX, 4
+ 1		Error return
+ 2		Normal return

- b) Error Conditions—an error return occurs when the characteristic of the given floating-point number is such that the fixed number would have no significant bits.
- c) Storage Required-25 cells (plus two cells of common storage).
- d) Time Required-0.056 millisecond, average
- e) Accuracy-26 significant bits.
- f) Exempt Symbol-FIX.
- g) Library Identification-LBR U1FXPT.

1.3.10 FIXED- TO FLOATING-POINT CONVERSION SUBROUTINE (U1FLPT) U1FLPT converts a number from fixed-point to floating-point format.

1.3.10.1 Input Requirement

The accumulator must contain the integral part of the fixed-point number, right-justified, and the multiplier-quotient must contain the fraction part of the fixed-point number upon entry into U1FLPT. The fixed-point number (FX) must satisfy the conditions: $2^{-35} < FX < 2^{37}$. If this number is an integer, the MQ must be cleared. If the magnitude of the number is less than one (the number is a fraction), the AC_{Q,P,1-35} must be cleared.

1.3.10.2 Output Requirements

The accumulator contains the converted floating-point number upon exit from U1FLPT.

1.3.10.3 Method

The 72-bit argument (AC $_{\rm Q,\,P,\,1-35}$ and MQ $_{\rm 1-35}$) is divided into two 27-bit bytes and an 18-bit byte. If the high-order byte is zero, it is temporarily stored as a normal floating-point zero. If the high-order byte is nonzero, it is given a characteristic of 165 and temporarily stored.

The second byte is examined and, if nonzero, is given a characteristic of 138 and temporarily stored. The low-order byte is right-justified and given a characteristic of 120. The three bytes now in normalized floating-point format are added, giving the normalized floating-point result.

1.3.10.4 Usage

a) Calling Sequence:

Location	<u>Operation</u>	Address, Tag, Decrement	
alpha	TSX	FLOAT, 4	
+ 1		Normal return	

- b) Error Conditions—none.
- c) Storage Required—21 cells (plus two cells of common storage).
- d) Time Required-0.071 millisecond, average
- e) Accuracy-26 significant bits.
- f) Exempt Symbol-FLOAT.
- g) Library Identification—LBR U1FLPT.

1.3.11 DOT PRODUCT SUBROUTINE (U3DOTP)

U3DOTP computes the dot (inner) product of two, real 3-dimensional vectors.

1.3.11.1 Input Requirements

The three components for each of the two operand vectors must occupy three consecutive locations in core storage in corresponding order and must be expressed in single-precision floating-point format. The first location of the components of one vector must be stored in LV1, the address of the second word of the calling sequence; the first location of the components of the second vector must be stored in LV2, the decrement of the second word of the calling sequence upon entry to U3DOTP. The program which uses U3DOTP provides locations designated by LV1 and LV2.

1.3.11.2 Output Requirements

The accumulator contains the floating-point scalar product upon exit from U3DOTP.

1.3.11.3 Method

If the components of \vec{v}_1 and \vec{v}_2 are \vec{v}_{1a} + \vec{v}_{1b} + \vec{v}_{1c} and \vec{v}_{2a} + \vec{v}_{2b} + \vec{v}_{2c} , the dot product is defined:

$$\vec{v}_1$$
 . \vec{v}_2 = \vec{v}_{1a} . \vec{v}_{2a} + \vec{v}_{1b} . \vec{v}_{2b} + \vec{v}_{1c} . \vec{v}_{2c}

1.3.11.4 Usage

a) Calling Sequence:

Location	Operation	Address, Tag, Decrement
alpha	TSX	U3DOT, 4
+ 1	PZE	LV1,, LV2
+ 2		Normal return

- b) Error Conditions—none.
- c) Storage Required-18 cells (plus two cells of common storage).
- d) Time Required:

Average 0.102 millisecond

Maximum 0.137 millisecond

- e) Accuracy-26 significant bits.
- f) Exempt Symbol—U3DOT.
- g) Library Identification-LBR U3DOTP.

1.3.12 CROSS PRODUCT SUBROUTINE (U3XPRO)

U3XPRO computes the cross (outer) product of two, real 3-dimensional vectors.

1.3.12.1 Input Requirement

The three components for each of the two openend vectors must occupy three consecutive locations in core storage in corresponding order and must be expressed in single-precision, floating-point format. The first location of the components of one vector must be stored in LV1, the address of the second word of the calling sequence; the first location of the components of the second vector must be stored in LV2, the decrement of the second word of the calling sequence upon entry to U3XPRO. The program which uses U3XPRO supplies the locations designated by LV1 and LV2.

1.3.12.2 Output Requirements

The three components of the vector cross product are stored in three consecutive core storage locations in the same order as that of the operand vectors upon exit from U3XPRO. The location in which the first component is stored, LV3, is contained in the address of the third word of the calling sequence and is supplied by the program which uses U3XPRO.

1.3.12.3 Method

If the components of \vec{v}_1 and \vec{v}_2 are \vec{v}_{1a} + \vec{v}_{1b} + \vec{v}_{1c} and \vec{v}_{2a} + \vec{v}_{2b} + \vec{v}_{2c} , the cross product is defined:

$$\vec{V}_{3a} = \vec{V}_{1b} (\vec{V}_{2c}) - \vec{V}_{1c} (\vec{V}_{2b})$$

$$\vec{V}_{3b} = \vec{V}_{1c} (\vec{V}_{2a}) - \vec{V}_{1a} (\vec{V}_{2c})$$

$$\vec{V}_{3c} = \vec{V}_{1a} (\vec{V}_{2b}) - \vec{V}_{1b} (\vec{V}_{2a})$$

1.3.12.4 Usage

Location	Operation	Address, Tag, Decrement
alpha	TSX	U3XPR, 4
+ 1	PZE	LV1,,LV2
+ 2	PZE	LV3
+ 3		Normal return

- b) Error Conditions-none.
- c) Storage Required-33 cells (plus two cells of common storage).
- d) Time Required:

Average

0.199 millisecond

Maximum

0.242 millisecond

- e) Accuracy—26 significant bits.
- f) Exempt Symbol—U3XPR.
- g) Library Identification-LBR U3XPRO.

1.3.13 MATRIX MULTIPLICATION SUBROUTINE (U3MATM)

U3MATM computes the product of two matrices.

1.3.13.1 Input Requirements

The two input matrices must be rectangular and must each be stored in consecutive core storage locations by rows. The address of the second and third words of the calling sequence must contain the first locations, respectively, of the two input matrices, A and B. The address of the fourth word of the calling sequence must contain the first location of the matrix product, C. The decrement of the second, third and fourth words must contain the matrix dimensions L, M and N, respectively, which define the dimensions of the three matrices in the following manner: if A is L x M and B is M x N, then C is L x N.

1.3.13.2 Output Requirements

The matrix product is stored by rows in L x N consecutive core storage locations, beginning with the location specified by C.

1.3.13.3 Method

U3MATM uses the standard row-by-column multiplication to obtain the matrix product elements.

1.3.13.4 Usage

a) Calling Sequence:

Location	Operation	Address, Tag, Decrement
alpha	TSX	U3MAT, 4
+ 1	PZE	A ,, L
+ 2	PZE	В,, М
+ 3	PZE	C ,, N
+ 4		Normal return

- b) Error Conditions—none.
- c) Storage Required-81 cells.
- d) Time Required-0.31 millisecond, average
- e) Accuracy-26 significant bits.
- f) Exempt Symbol—U3MAT.
- g) Library Identification-LBR U3MATM.

1.3.14 STATION COORDINATE CONVERSION SUBROUTINE (UA1LSC)

UA1LSC converts an inertial position vector into values of range, azimuth and elevation. The flow chart for UA1LSC is shown in Figure 1-1.

1.3.14.1 Input Requirements

The address of the second word of the calling sequence must contain the location of a 5-cell input block; the decrement of the second word must contain the first location of a 12-cell block which contains the output. The 5-cell input block contains:

Location	Contents
LCINP	Time of observation (minutes in address, seconds in decrement) in GMT for the following inertial position vector.

MC 63-4

Location	Contents
+ 1	X component of spacecraft position, floating-point
+ 2	Y component of spacecraft position, floating-point
+ 3	Z component of spacecraft position, floating-point
+ 4	Location of the station characteristics block for the station.

UA1LSC must have access to the following programs: U1SQRT, U1ATAB and U1SICO.

1.3.14.2 Output Requirements

UA1LSC fills a 12-cell output block as shown below:

Location	Contents
LCOUT	Range (R), floating-point Mercury units
+ 1	Azimuth angle (A), floating-point radians
+ 2	Elevation angle (E), floating-point radians
+ 3	-sin λ
+ 4	cos λ
+ 5	0
+ 6	$-\cos\lambda \sin\varphi$
+ 7	$-\sin\lambda \sin \varphi$
+ 8	$\cos \phi$
+ 9	$\cos \lambda \cos \phi$
+ 10	$\sin \lambda \cos \phi$
+ 11	$\sin \phi$

The last nine cells contain the three rows of the M matrix. The angles are defined in the following subsection.

1.3.14.3 Method

UA1LSC uses the following equations to compute range, azimuth and elevation:

$$\lambda = \lambda_0'' + \omega_e t$$

$$\begin{pmatrix} X'' \\ Y' \\ Z'' \end{pmatrix} = \begin{pmatrix} -\sin \lambda & \cos \lambda & 0 \\ -\cos \lambda & \sin \phi & -\sin \lambda & \sin \phi & \cos \phi \\ \cos \lambda & \cos \phi & \sin \lambda & \cos \phi & \sin \phi \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \begin{pmatrix} R_s \sin(\phi - \phi') \\ -R_s \cos(\phi - \phi'') \end{pmatrix}$$

$$R = \sqrt{(X'')^2 + (Y')^2 + (Z')^2} \qquad \sin E = \frac{Z''}{R}$$

$$A = \arctan \frac{X'}{Y'}$$

$$\cos E = \sqrt{1 - \sin^2 E}$$

where:

 $E = \arctan \frac{\sin E}{\cos E}$

 λ = longitude of the station λ_0'' = longitude of station at reference time t = time (in minutes) since reference time ω_e = rotation of earth, radians/minute = 0.00437526905 X, Y, Z = given inertial spacecraft position coordinates R_S = geocentric radius of earth at station, Mercury units φ = geodetic latitude of station φ'' = geocentric latitude of station

1.3.14.4 Usage

Location	Operation	Address, Tag, Decrement
alpha	TSX	A1LSC, 4
+ 1	PZE	LCINP,, LCOUT

Location	Operation	Address, Tag, Decrement
+ 2		Error return
+ 3		Normal return

- b) Error Conditions—the decrement of A1LSC contains the source of the error return—1, 2 or 4 indicate error returns from U1SQRT, U1ATAB or U1SICO, respectively. There is no error return from UA1LSC itself.
- c) Storage Required—209 cells.
- d) Time Required (includes execution of U1SICO, U1SQRT and U1ATAB, each twice): 4.9 millisecond, average
- e) Accuracy—the range (R) is accurate to 26 significant bits; the azimuth (Z) and elevation (E) angles are both accurate to seven significant digits.
- f) Exempt Symbol—A1LSC.
- g) Library Identification—LBR UA1LSC.

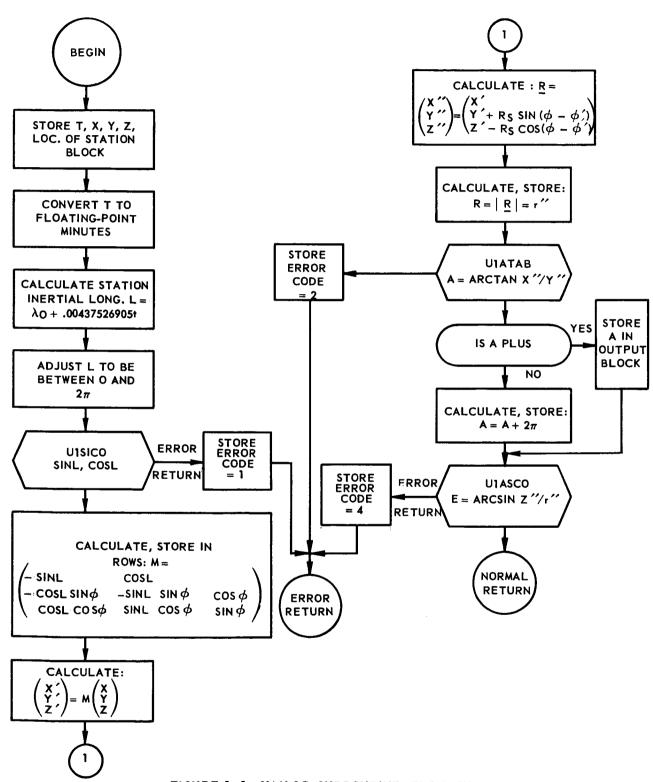


FIGURE 1-1. UAILSC SUBROUTINE FLOW CHART

1.3.15 SIX-POINT LAGRANGIAN INTERPOLATION SUBROUTINE (U7INTP)

U7INTP performs a 6-point Lagrangian interpolation to determine intermediate values of the radius or radius and velocity vectors. The flow chart for U7INTP is shown in Figure 1-2.

1.3.15.1 Input Requirements

The following conditions apply for all entries to U7INTP:

- a) All vector components must be expressed in floating-point format.
- b) The address (TOBMN) and the decrement (TOBSC) of the second word of the calling sequence must contain the minutes and seconds, respectively, of the time for interpolation.
- c) The address (LOCNI) and the decrement (LCOUT) of the third word in the calling sequence must contain, respectively, the location of the numerical integration table to be used for interpolation and the location for storing the generated output block of vector components.

The fourth word (Alpha + 3) of the calling sequence correlates the numerical integration input table with the interpolation output required, as shown below:

a) Combinations for Interpolation:

Interpolation	Numerical Integration Table	Address, Tag, Decrement $(\alpha + 3)$
Radius	Radius	0,,0
Radius	Radius and velocity	0,,1
Radius and velocity	Radius and velocity	1,,1

b) Format of Numerical Integration Table:

Location	Contents
LOCNI	Time increment of table (0 = minutes, 1 = seconds)
+ 1	Integration output interval
+ 2	Time tag of first vector in table
+ 3	Time tag of last vector in table
+ 4	
+ 5	
+ 6	Components of first radius vector
+ 7	

Location	Contents
+ 8	Components of first velocity vector, if table contains
+ 9	radius and velocity; components of second radius vec-
+ 10	tor if table contains only radius values.
etc.	

U7INTP may use either a "minutes" integration table if the time of interpolation is an integral number of minutes, or a "seconds" integration table if the time is an integral number of seconds.

c) Special Case, Extract Anchor Point—with the tag of the second word of the calling sequence (T) equal to 1, U7INTP extracts the radius and velocity vectors for the time specified, TOBMN, from the specified numerical integration table. The table must contain both radius and velocity vectors (the decrement of Alpha + 3 must be 1) and the table must contain components of the vectors for 1-minute intervals (the contents of location LOCNI must be zero).

1.3.15.2 Output Requirements

The output from U7INTP depends upon the conditions established at entry.

a) Interpolation Case:

Location	Contents		
LCOUT	\vec{r}_{x}		
+ 1	ry Interpolated components of radius vector		
+ 2	$\vec{\mathbf{r}}_{\mathbf{z}}$		
+ 3	$\vec{v}_{_{\mathbf{X}}}$		
+ 4	vy Interpolated components of velocity vector, if velocity interpolation is required		
+ 5	\vec{v}_z		

b) Anchor Point Extraction:

Location	Contents		
LCOUT	TOBMN ("Time of Observation" in minutes)		
+ 1	$\vec{r}_{\mathbf{x}}$		
+ 2	r Extracted radius vector		
+ 3	$\vec{\mathbf{r}}_{\mathbf{z}}$		
+ 4	$\vec{\mathbf{v}}_{\mathbf{x}}$		
+ 5	v _y Extracted velocity vector		
+ 6	$\vec{\mathbf{v}}_{\mathbf{z}}$		

1.3.15.3 Method

a) Interpolation—U7INTP uses a 6-point Lagrangian interpolation. The numerical integration table must contain the three points prior to and after the input time (TOB). The interpolated radius (or vector) component \vec{r} is computed from the formula:

$$\vec{r} = \sum_{i=-2}^{3} L_{i} \vec{r}_{i}$$

where \vec{r}_{-2} , \vec{r}_{-1} and \vec{r}_0 are the vector components for the three times immediately prior to the input time; \vec{r}_1 , \vec{r}_2 and \vec{r}_3 are the vector components for the three times immediately following the input time; and L_i is defined as:

$$L_{-2} = -\frac{1}{120} (t+1) (t) (t-1) (t-2) (t-3)$$

$$L_{-1} = +\frac{1}{24} (t+2) (t) (t-1) (t-2) (t-3)$$

$$L_{0} = -\frac{1}{12} (t+2) (t+1) (t-1) (t-3)$$

$$L_{1} = + \frac{1}{12} (t + 2) (t + 1) (t) (t - 2) (t - 3)$$

$$L_{2} = -\frac{1}{24} (t + 2) (t + 1) (t) (t - 1) (t - 3)$$

$$L_{3} = + \frac{1}{120} (t + 2) (t + 1) (t) (t - 1) (t - 2)$$

and t is the ratio of the time increment (time of \vec{r} minus time of \vec{r}_0) to the interval between any two adjacent corresponding vector components in the numerical integration table. Therefore, t < 1.

b) Anchor Point Extraction—U7INTP searches the given numerical integration table for the required values and places them in the assigned output location.

1.3.15.4 Usage

Location	Operation	Address, Tag, Decrement
alpha	TSX	UINTP, 4
alpha + 1	PZE	TOBMN,, TOBSC
alpha + 2	PZE	LOCNI, T, LCOUT
alpha + 3	PZE	(Output),, (Input)
alpha + 4		Error return
alpha + 5	•	Normal return

- b) Error Conditions—an error return occurs if sufficient vector components are not available in the numerical integration table to perform a 6-point interpolation. The contents of the accumulator indicate which part of the numerical integration table prohibits interpolation: zero indicates insufficient vectors at the beginning; a 1 indicates insufficient vectors at the end. An error return also occurs if, in the anchor point case, a numerical integration table with second intervals is specified.
- c) Storage Required-258 cells (plus five additional cells of common storage).

d) Time Required:

Interpolation	<u>IBM 7094</u>
\vec{r} only	2.81 milliseconds
$\overrightarrow{\mathbf{r}}$ and $\overrightarrow{\mathbf{v}}$	3.76 milliseconds
Anchor Point	0.26 millisecond

- e) Accuracy-26 significant bits.
- f) Exempt Symbol-UINTP.
- g) Library Identification—LBR U7INTP.

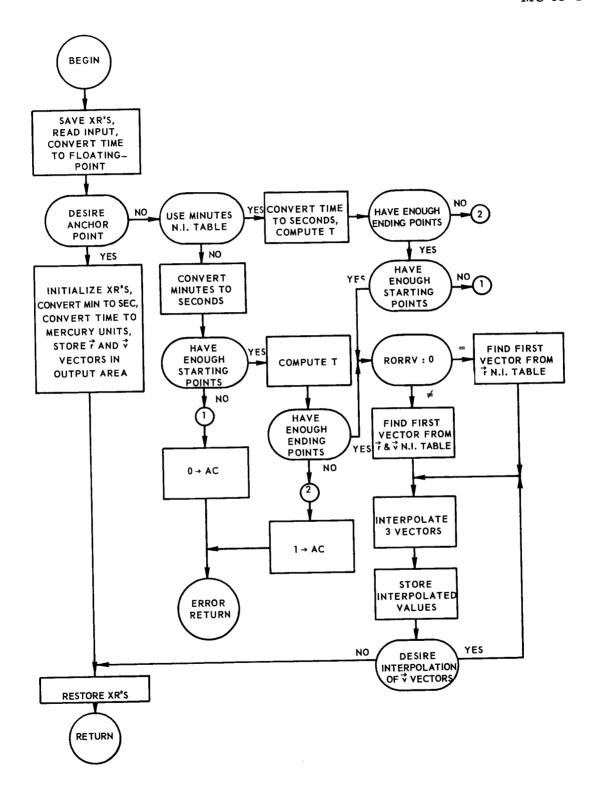


FIGURE 1-2. U7INTP SUBROUTINE FLOW CHART

1.3.16 VECTOR MAGNITUDE SUBROUTINE (U3VMAG)

U3VMAG generates the magnitude of a given vector.

1.3.16.1 Input Requirements

The three components for the vector must be stored in three consecutive core storage locations and must be expressed in floating-point format. The first location of the components of the vector must be stored in the address of the second word of the calling sequence, as designated by LV1, upon entry into U3VMAG. The program must have access to the square root program, U1SQRT.

1.3.16.2 Output Requirements

The accumulator contains the floating-point magnitude of the given vector components upon exit from U3VMAG.

1.3.16.3 Method

If the components of the vector, V, are $V_a + V_b + V_c$, the magnitude of the vector is defined:

$$V = \sqrt{V_a^2 + V_b^2 + V_c^2}$$

1.3.16.4 Usage

a) Calling Sequence:

Location	Operation	Address, Tag, Decrement
alpha	TSX	VMAG, 4
alpha + 1	PZE	LV1
alpha + 2		Error return
alpha + 3		Normal return

b) Error Condition—an error return is an almost certain sign of machine malfunction. An even return from U3VMAG is the result of an error return from the subroutine U1SQRT. Floating-point underflow or overflow is possible if the characteristic, C, of any of the components of the given vector fails to satisfy the relation: 63 < C < 192.

- c) Storage Required-19 cells (plus one cell of common storage).
- d) Time Required (includes execution of U1SQRT):

Average

0.130 millisecond

Maximum

0.292 millisecond

- e) Accuracy-26 significant bits.
- f) Exempt Symbol-VMAG.
- g) Library Identification—LBR U3VMAG.

1.3.17 VECTOR TRIPLE CROSS-PRODUCT SUBROUTINE (U3VPRO)

U3VPRO computes the vector cross-product of three given vectors.

1.3.17.1 Input Requirements

The three components of each of the three input vectors must occupy three consecutive locations in core storage upon entry into U3VPRO. The components for each vector must be arranged in corresponding order. If the cross-product is defined as $\vec{W} \times (\vec{U} \times \vec{V})$, the location of the first component of \vec{W} must be stored in the address of the second word of the calling sequence (LOC \vec{W}_a), the location of the first component of \vec{U} must be stored in the decrement of the second word of the calling sequence (LOC \vec{U}_a), and the location of the first component of \vec{V} must be stored in the address of the third word of the calling sequence (LOC \vec{V}_a).

The first location of the three cells, where U3VPRO stores the resultant vector \vec{Y} components, must be stored in the decrement of the third word of the calling sequence (LOC \vec{Y}_2). All values must be in floating-point format.

1.3.17.2 Output Requirements

The three components of the resultant vector are stored in three consecutive core storage locations beginning with the $LOCY_a$.

1.3.17.3 Method

If
$$\vec{Y} = \vec{W} \times (\vec{U} \times \vec{V})$$
, then $\vec{Y} = (\vec{W} \cdot \vec{V}) \vec{U} - (\vec{W} \cdot \vec{U}) \vec{V}$.

1.3.17.4 Usage

a) Calling Sequence:

Location	<u>Operation</u>	Address, Tag, Decrement
alpha	TSX	VPRO, 4
+ 1	PZE	$\mathtt{LOC}ec{ extbf{w}}_{ extbf{a}}$, , $\mathtt{LOC}ec{ extbf{U}}_{ extbf{a}}$
+ 2	PZE	$\mathtt{LOC} \vec{\mathrm{v}}_{\mathbf{a}}$, $\mathtt{LOC} \vec{\mathrm{v}}_{\mathbf{a}}$
+ 3		Normal return

- b) Error Conditions—none.
- c) Storage Required-80 cells.
- d) Time Required-0.490 millisecond, average
- e) Accuracy-eight significant digits.
- f) Exempt Symbol-VPRO.
- g) Library Identification-LBR U3VPRO.

1.3.18 UNIT VECTOR SUBROUTINE (U3UNTV)

U3UNTV generates the unit vector of a given vector.

1.3.18.1 Input Requirements

The three components of the given vector must occupy three consecutive locations in core storage and must be expressed in floating-point format. The first location of the three components must be stored in LV1, the address of

the second word of the calling sequence. The program which uses U3UNTV supplies the location designated by LV1. U3UNTV must have access to the square root program, U1SQRT.

1.3.18.2 Output Requirements

The three components of the generated unit vector are stored in three consecutive core storage locations in the same order as that of the given vector. The location in which the first component is stored, LV2, is contained in the decrement of the second word of the calling sequence and is supplied by the program which uses U3UNTV.

1.3.18.3 Method

The vector magnitude, V, of the vector \vec{V} is computed from the components \vec{V}_a , \vec{V}_b and \vec{V}_c . The unit vector components are computed by dividing the vector component by the vector magnitude. Therefore:

$$\vec{v}_a = \frac{\vec{v}_a}{\vec{v}} \qquad \vec{v}_b = \frac{\vec{v}_b}{\vec{v}} \qquad \vec{v}_c = \frac{\vec{v}_c}{\vec{v}}$$

1.3.18.4 Usage

Location	Operation	Address, Tag, Decrement
alpha	TSX	UNITV, 4
+ 1	PZE	LV1,,LV2
+ 2		Error return
+ 3		Normal return

- b) Error Conditions—an error return is a logical impossibility and implies a machine malfunction.
- c) Storage Required—30 cells (plus two cells of common storage).

d) Time Required (includes execution of U1SQRT):

Average

0.374 millisecond

Maximum

0.406 millisecond

- e) Accuracy—26 significant bits.
- f) Exempt Symbol—UNITV.
- g) Library Identification-LBR U3UNTV.

1.3.19 COMPUTATION OF ELLIPTIC MOTION DURING LAUNCH PHASE SUBROUTINE (C9RVTH)

C9RVTH, called from the SOS library tape by LBR U7RVTH, predicts the position and velocity of the spacecraft under the assumption of elliptic motion with no drag or oblateness perturbations.

Given the position and velocity at some time t, C9RVTH computes the position and velocity at time $t + \Delta t$ or at a specified height above the earth.

The flow chart for C9RVTH is shown in Figure 1-3.

1.3.19.1 Input Requirements

C9RVTH may be entered at either of two places, C9RVTH or C9RVT2, depending on whether new orbit elements must be computed. The input which must be provided to obtain the desired output is summarized below:

a) Entry at C9RVTH.

<u>AC</u>	MQ	<u>Input</u>	Output
plus	minus	\vec{r} , \vec{v} , Δt	\overrightarrow{r} , and \overrightarrow{v}
zero	minus	\vec{r} , \vec{v} , h_s , R_s	\vec{r} , \vec{v}' and Δt
minus	minus	r, v, h, Rs	\vec{r} , \vec{v} , γ and Δt
plus	plus	\vec{r} , \vec{v} , Δt	→¹ r
minus or zero	plus	\vec{r} , \vec{v} , h_s , R_s	\vec{r} and Δt

b) Entry at C9RVT2. (Prior entry must have been made at C9RVTH and the initial \vec{r} and \vec{v} and the corresponding orbit elements must be available in core.)

AC	MQ	Input	Output
plus	minus	Δt	$\overrightarrow{\mathbf{r}}'$ and $\overrightarrow{\mathbf{v}}'$
zero	minus	h _s , R _s	\vec{r} , \vec{v} and Δt
minus	minus	h _s , R _s	\vec{r} , \vec{v} , γ and Δt
plus	plus	Δt	→t T
minus or zero	plus	h _s , R _s	$\overset{\rightarrow}{\mathbf{r}}$ and $\Delta \mathbf{t}$

All values are Mercury units. The computed Δt is assumed positive. Eccentricity, e, is set = 1 if it initially exceeds unity.

C9RVTH must have access to the U1SQRT, U1ATAB, U1ATNA, U3DOTP, U3VMAG, and C9ASKE subroutines.

The error returns from all of the above subroutines are NOPs.

The locations of the above input and output quantities are specified in the calling sequence to C9RVTH (see subparagraph 1.3.19.4a).

1.3.19.2 Output Requirements

See above. If Δt was computed, it will be in the AC in Mercury units upon exit from C9RVTH.

1.3.19.3 Method

$$\mathbf{r} = |\vec{\mathbf{r}}| \text{ and } \mathbf{v} = |\vec{\mathbf{v}}|$$

$$e cos E = rv^2 = -1$$

$$a = \frac{r}{1 - e \cos E}$$

$$n = \sqrt{\frac{1}{a^3}}$$

$$e \sin E = \frac{\overrightarrow{r} \cdot \overrightarrow{v}}{a}$$

$$e = \sqrt{(e \sin E)^2 + (e \cos E)^2}$$

If $e < e_{\mbox{critical}}$, the motion is nearly circular, and the calculation of the new position and velocity is not made. A special exit is made from C9RVTH.

E = arc tan
$$\left(\frac{e \sin E}{e \cos E}\right)$$
, $-\pi < E \le \pi$
M = E - e sin E.

If h_s is given, compute $\vec{r}' = R_s + h_s$. (If r' > a (1 + e), the specified height, h_s , is greater than the apogee height and therefore cannot be attained, and a special exit is made from C9RVTH).

e cos E' =
$$\frac{\mathbf{a} - \mathbf{r'}}{\mathbf{a}}$$

e sin E' = $-\sqrt{\mathbf{e}^2 - (\mathbf{e} \cos \mathbf{E'})^2}$
 $\mathbf{M'} = \mathbf{E'} - \mathbf{e} \sin \mathbf{E'}$
 $\Delta \mathbf{t} = \frac{\mathbf{M'} - \mathbf{M}}{\mathbf{n}}$

If Δt is given, compute M' = M + n (Δt). Solve Keplers' equation M' = E' - e sin E' for E' using the subroutine C9ASKE.

$$f = \frac{-e \cos E + \cos E' \cos E + \sin E' \sin E}{1 - e \cos E}$$

$$g - \frac{1}{n} [(\cos E - e) \sin E' - (\cos E' - e) \sin E]$$

$$r' = f\vec{r} + g\vec{v}.$$
Either $r' = a(1 - e \cos E')$

$$v' = \sqrt{\frac{2 - 1}{r' - a}}$$

$$\tan \gamma' = \sqrt{\frac{e \sin E'}{1 - e^2}}$$

$$\gamma' = \arctan (\tan \gamma').$$
Or $f' = n \left[\frac{\sin E \cos E' - \sin E' \cos E}{(1 - e \cos E') (1 - e \cos E)} \right]$

$$g' = \left[-\frac{\cos E' + \cos E' \cos E + \sin E' \sin E}{1 - e \cos E'} \right]$$

$$\vec{v}' = f' \vec{r} + g' \vec{v}.$$

1.3.19.4 Usage

a) Calling Sequence:

alpha	TSX	C9RVTH, 4	(orbit elements have NOT been calculated)
	or TSX	C9RVT2, 4	(orbit elements have been previously calculated)
+ 1	PZE	$L(\vec{r}), L(\vec{v})$	
+ 2	PZE	A,,B	
+ 3	PZE	L (r'),, C	
+ 4	Error return	(circular orb	oit)
+ 5	Error return	(cannot reach	n h _s)
+ 6	Normal return	n	

where the symbols in the calling sequence are defined as:

AC	$\underline{\mathbf{MQ}}$	<u>A</u>	<u>B</u>	<u>C</u>
plus	minus	L (△ t)	zero	$L(\overrightarrow{v})$
zero	minus	L (R _s)	L (h _s)	$L(\vec{v})$
minus	minus	$L(R_g)$	L (h _s)	$L(\overrightarrow{v})$

AC	MQ	<u>A</u>	<u>B</u>	<u>C</u>
zero or minus	plus	L (R _s)	L (h _s)	zero
plus	plus	L (△ t)	zero	zero

- and L (r) is the address of the first location of a 3-word block containing the components of the input position vector, in Mercury units.
 - L (v) is the address of the first location of a 3-word block containing the components of the input velocity vector, in Mercury units.
 - L (Δt) is the location containing the input quantity Δt , in Mercury units.
 - L(R_s) is the location containing the input quantity R_s, radius of the earth, in Mercury units.
 - $L(h_s)$ is the location containing the input quantity h_s , height of spacecraft above the earth, in Mercury units.
 - L (r) is the address of the first location of the 3-word block containing components of the computed position vector.
 - L (v) is the address of the first location of the 3-word block containing components of the computed velocity vector.
 - L (v) is the address of the first location of a 2-word block; the first word contains the magnitude of the computed velocity, the second word contains the computed flight path angle.
- b) Error conditions:
 - 1) A return to alpha + 4 indicates a circular orbit.
 - 2) A return to alpha + 5 indicates the orbit will not reach h_s .
 - 3) Error returns from subroutines used have been set to NOPs.
- c) Storage Required-380₁₀ locations, including constants and temporary storage, but excluding subroutines.
- d) Time Required (maximum number of IBM 7094 machine cycles, excluding subroutines)—1333₁₀.

- e) Accuracy-Seven significant decimal digits.
- f) Exempt symbols—C9RVTH and C9RVT2.
- g) Library Identification—LBR U7RVTH.

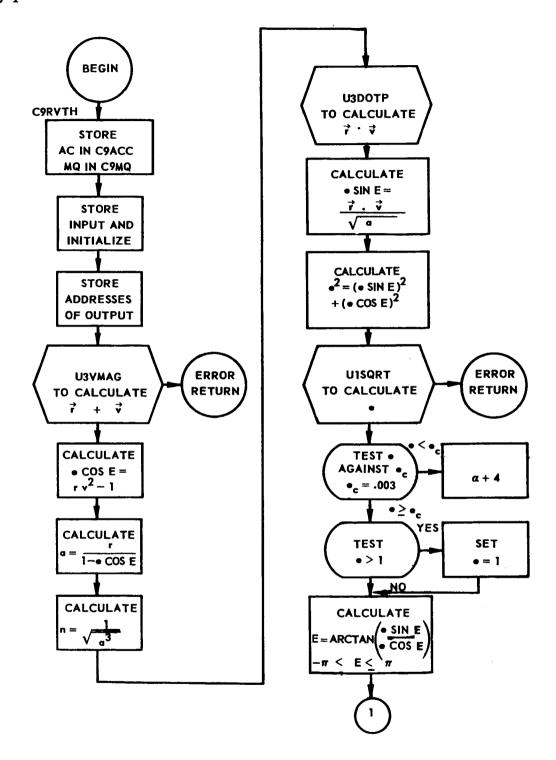


FIGURE 1-3. C9RVTH SUBROUTINE FLOW CHART (Sheet 1 of 4)

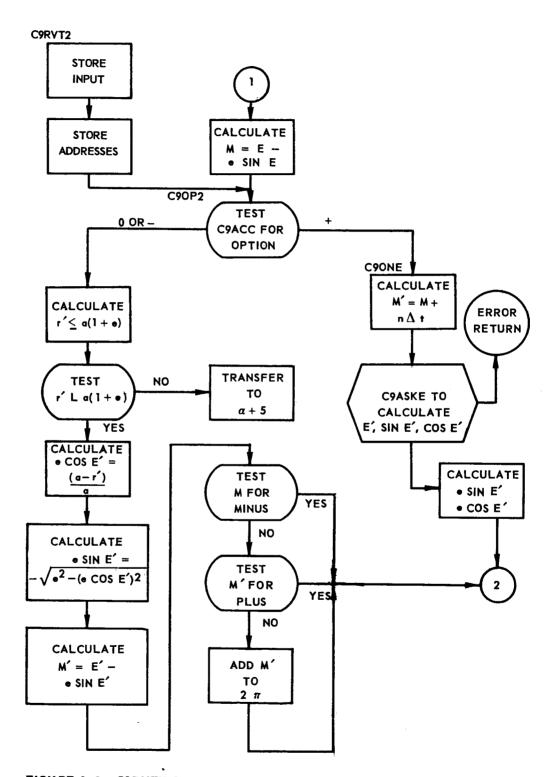


FIGURE 1-3. C9RYTH SUBROUTINE FLOW CHART (Sheet 2 of 4)

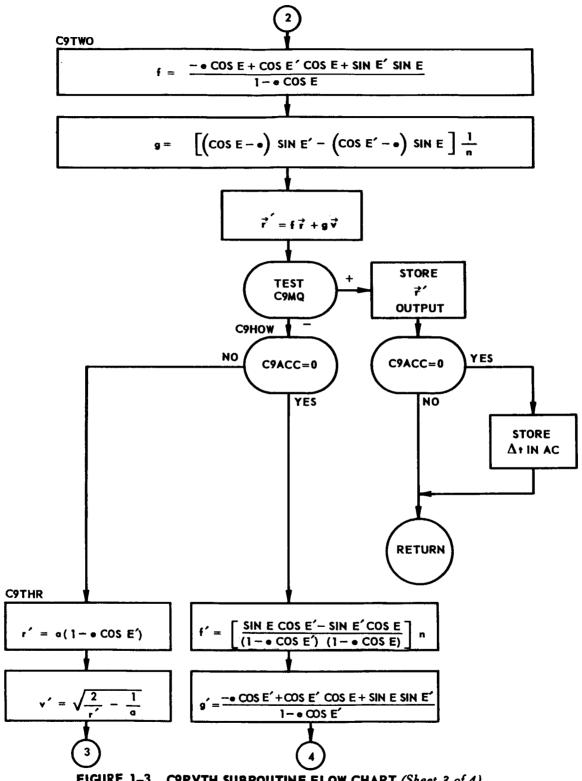


FIGURE 1-3. C9RYTH SUBROUTINE FLOW CHART (Sheet 3 of 4)

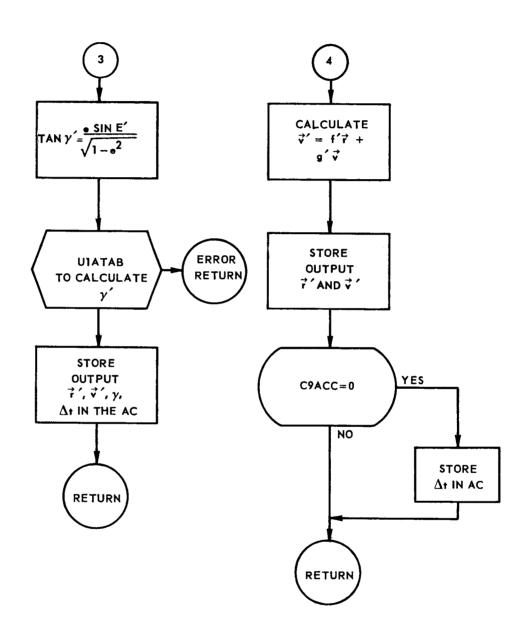


FIGURE 1-3. C9RVTH SUBROUTINE FLOW CHART (Sheet 4 of 4)

1.3.20 SOLUTION OF KEPLER'S EQUATION SUBROUTINE (C9ASKE)

C9ASKE, called from the SOS library tape by LBR U7ASKE, solves the equation $M = E - E \sin E$ for E, Sin E, and Cos E, given M and e.

1.3.20.1 Input Requirements

The eccentricity of orbital ellipse e, and the mean anomaly at time t, M, both in floating-point, must be available in two consecutive locations. The address of the first of these locations is specified in the address of the first word of the calling sequence to C9ASKE. C9ASKE must have access to the U1SICO subroutine.

1.3.20.2 Output Requirements

E, Sin E, Cos E, and the eccentric anomaly at time t, all in floating-point, are stored in a 3-word output block. The address of the first location of the block is specified in the decrement of the first word of the calling sequence to C9ASKE.

1.3.20.3 Method

$$E^{i+1} = E^{i} - \frac{E^{i} - e \sin(E^{i}) - M}{1 - e \cos(E^{i})}$$
.

Iteration is terminated when convergence to 0.0001 degrees (= 0.000017 radians) has been reached or when the maximum number of iterations has been performed. Presently this number is set to ten.

1.3.20.4 Usage

alpha	TSX	C9ASKE, 4
+ 1	PZE	L (input),, L (output)
+ 2		Error return
+ 3		Normal return

b) Error Conditions:

- A return to alpha + 2 with 1 in the decrement of location C9ASKE indicates that the AC overflow indicator or the divide check indicator on the IBM 7094 console was turned on during execution of C9ASKE.
- 2) A return to alpha + 2 with 2 in the decrement of location C9ASKE indicates an error return from the U1SICO subroutine.
- 3) A return to alpha + 2 with 3 in the decrement of location C9ASKE indicates that convergence was not accomplished in the maximum number of iterations (presently ten) allowed.
- c) Storage Required—104₁₀ including constants and temporary storage, but excluding the U1SICO subroutine.
- d) Time Required—(maximum number of IBM 7094 machine cycles, excluding the subroutine U1SICO): 1418₁₀.
- e) Accuracy-26 significant bits.
- f) Exempt symbol—C9ASKE.
- g) Library Identification-LBR U7ASKE.

1.3.21 SOS LIBRARY TAPE WRITER PROGRAM (LBRWR)

LBRWR prepares a library tape of utility subroutines for use with Mercury SOS. The flow chart for LBRWR is shown in Figure 1-4.

1.3.21.1 Input Requirements

The Library Tape Writer program requires a squoze tape on A5 as input. Each subroutine must be a separate file, and two adjacent EOF marks must be present to signify the end of the input tape. The squoze tape may be prepared by SOS by compiling the utility routines, in order, and stacking the squoze on tape (SSW #6 up). SOS currently writes the EOF between jobs automatically.

The input deck to prepare a squoze tape consists of:

- a) JOB card (columns 16-21 containing the symbol destined to be used with the LBR card to call in that subroutine)
- b) CPL card
- c) Symbolic deck of first utility subroutine
- d) Blank card
- e) (a) through (d) for each of the remaining subroutines
- f) PAUSE card

The current version of SOS expects a maximum of 26 utility routines.

1.3.21.2 Output Requirements

LBRWR produces a library tape on A4. The tape is composed of two files: the first file is empty; the second file contains one BCD ID record followed by binary records (maximum of 256 words each) for the utility routines.

1.3.21.3 Method

LBRWR writes an EOF followed by a 1-word record in BCD on the output tape. This record identifies the tape as a library tape. LBRWR then edits the utility subroutines and writes the edited squoze information as the remainder of the second file on the output tape. Upon sensing a double EOF on the input tape, LBRWR writes an EOF and rewinds the output tape.

1.3.21.4 Usage (Operator's Procedures):

- a) Ready SOS on A1.
- b) Ready blank tapes on A2, A4, B1, and B2.
- c) Ready input squoze tape on A5.
- d) Ready LBRWR program in card reader.
- e) Set sense switch #1 down.
- f) Press LOAD TAPE button.
- g) Program final stop is at 1402₈, with the library tape rewound on A4.

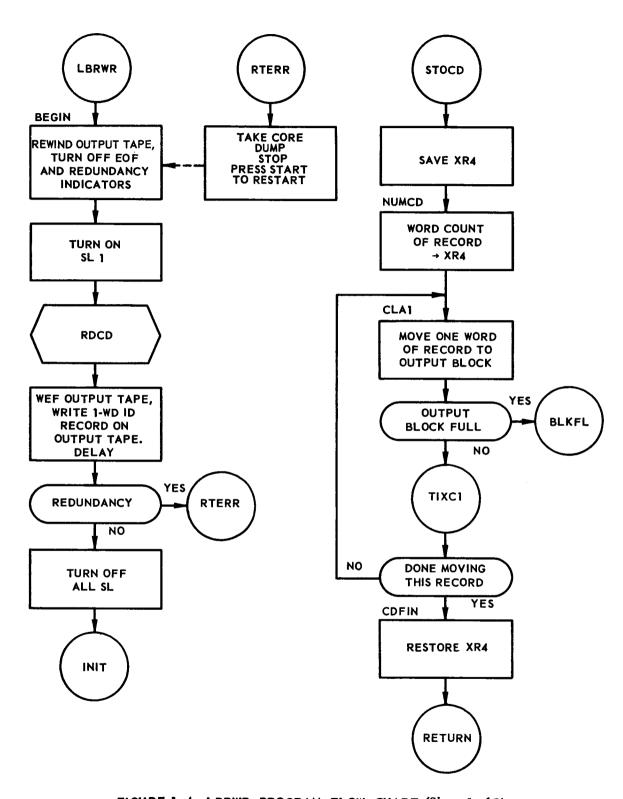


FIGURE 1-4. LBRWR PROGRAM FLOW CHART (Sheet 1 of 5)

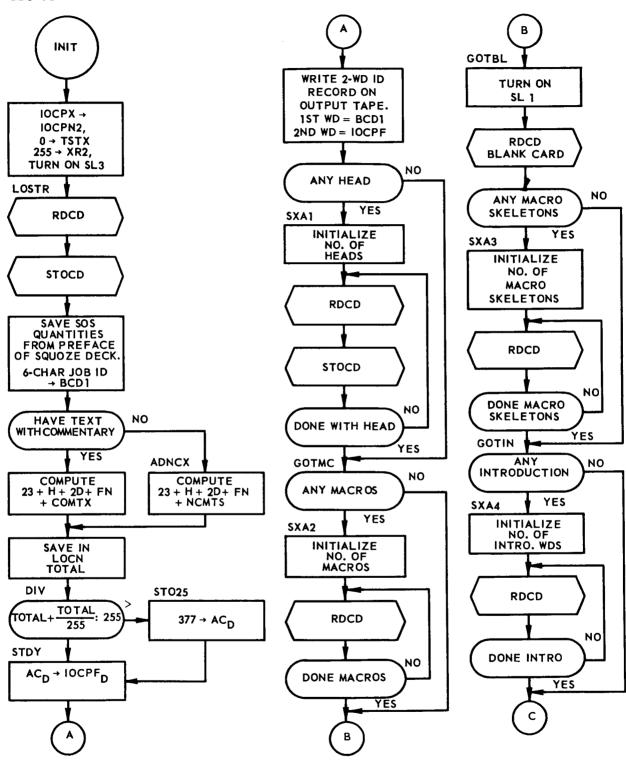


FIGURE 1-4. LBRWR PROGRAM FLOW CHART (Sheet 2 of 5)

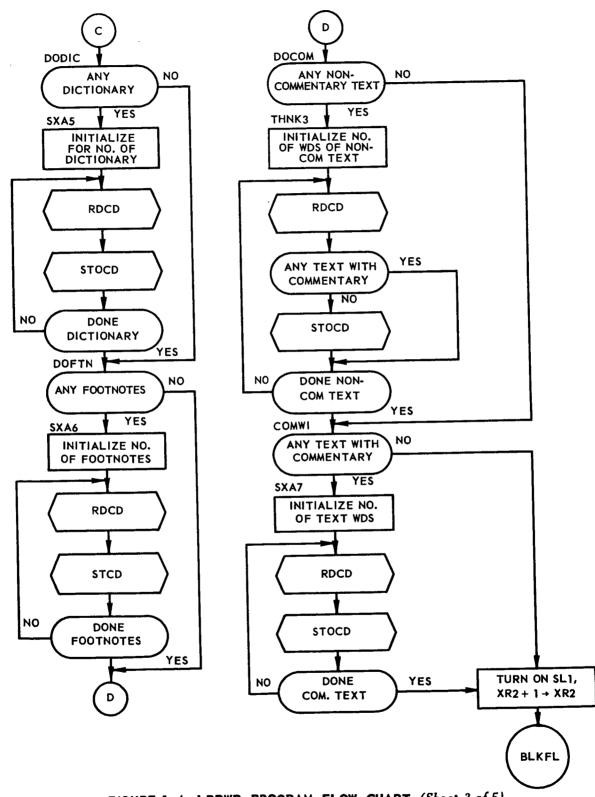


FIGURE 1-4. LBRWR PROGRAM FLOW CHART (Sheet 3 of 5)

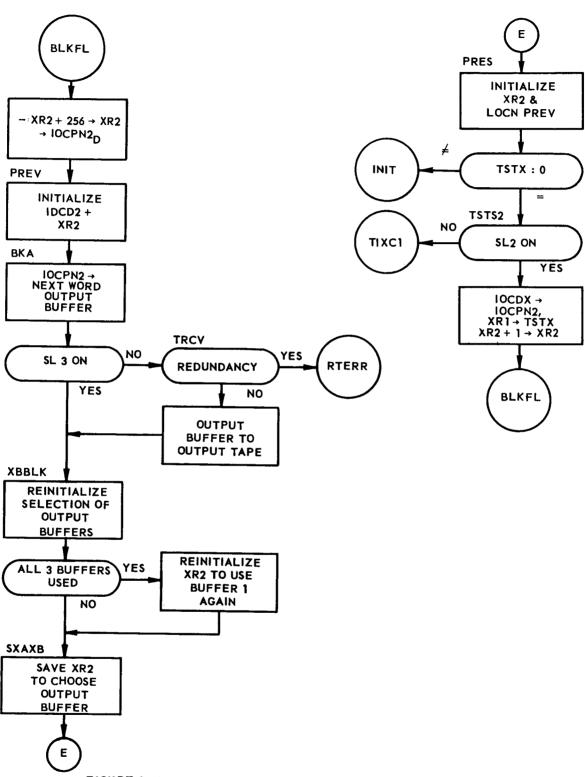


FIGURE 1-4. LBRWR PROGRAM FLOW CHART (Sheet 4 of 5)

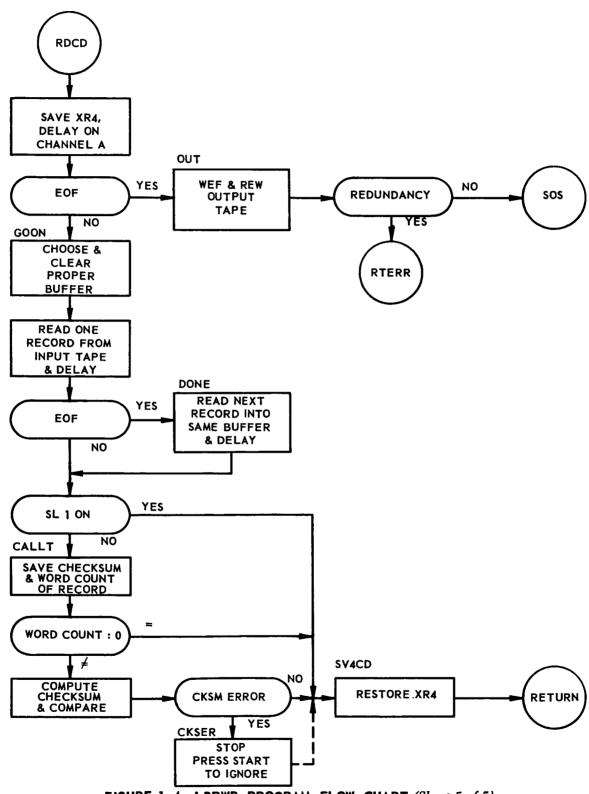


FIGURE 1-4. LBRWR PROGRAM FLOW CHART (Sheet 5 of 5)

Section 2

SIMULATION PROGRAMS

Simulation programs pretest the Project Mercury Programming System in an environment which approximates a real-time mission. These programs are divided into three categories: (1) data preparation programs, (2) simulated input/output control program (SIC), and (3) open and closed loop programs for project personnel training.

2.1 OBSERVER PROGRAM (OBSER)

Observer produces a set of pure radar observations of the Mercury space-craft. These observations are degraded to simulate random radar and transmission errors and are then reformatted by the Selector and Shred programs. Following this, the observations are used as input to the simulated input/output control (SIC) program which feeds data to Mercury monitor in simulated real time.

The flow chart for Observer is shown in Figure 2-1.

2.1.1 Input Requirements

A functional description of the possible inputs to Observer is given below.

- a) Station Characteristics Tape (prepared by U0STCH)—contains position of site and type of radar.
- b) Space Technology Laboratories (STL) Tape—loaded from column binary cards containing position/velocity vectors in B-GE coordinates for a simulated launch. This powered flight data is provided by STL.
- c) Cards (read from the on-line reader)—specify all the parameters for a run. The sequence of cards is:
 - 1) Four parameter cards containing spacecraft characteristics.
 - 2) An identification card.
 - 3) A card, or cards, specifying the manner in which a table of position/velocity vectors is to be generated. There are four ways in which this can be done:
 - (a) Converting STL launch data in B-GE coordinates to inertial coordinates.
 - (b) Numerical integration (for free flight) starting from a given position/velocity vector.
 - (c) Extending a previously generated table by firing posigrade rockets at a specified time and then integrating.
 - (d) Extending a previously generated table by firing retrorockets at a specified time and then integrating.

- 4) A set of station cards, each identifying a particular radar site.

 These cards control the reading of the Station Characteristics tape so positional data from each site can be obtained. Radar pointing data for each station card is calculated assuming that the spacecraft traces the path described in the table of position vectors.
- 5) An END card.

A typical set of cards for obtaining launch, orbit and reentry data for the two Bermuda radars would be:

- a) Four parameter cards.
- b) Identification card, identifying the run.
- c) STL card indicating use of STL tape data for powered flight.
- d) Station card for Bermuda Verlort radar.
- e) Station card for Bermuda AN/FPS-16 radar.
- f) New parameter card(s), if desired.
- g) Execute posigrade (XP) card for extending flight by firing posigrade rockets and going into free flight.
- h) New station card(s), if desired.
- i) New parameter card(s), if desired.
- j) Execute retrograde (XR) card for extending flight by firing retrograde rockets and returning to free flight for the reentry phase.
- k) New station card(s), if desired.
- 1) END card.

2.1.2 Output Requirements

A listing of the possible Observer outputs includes:

a) A binary tape (B5) containing in each record a table of position/velocity vectors. If STL data is used, the first record contains an edited version of the B-GE data; all other records contain data in inertial coordinates. There is one record for each table generated in the Observer program.

- b) A BCD tape (A8) for printing off line the radar observations made at each station, a table of position/velocity vectors, and a table of subvehicle positions.
- c) A binary tape (B6) for input to the Selector program, containing the radar observations made at each station. This is the only optional output and is not used unless radar reports are actually calculated.

2.1.3 Method

Observer first calculates a set of position/velocity vectors and then (using only the position vectors) calculates radar pointing data from specified stations to the path described in the vector table.

As indicated by the selection of cards, the vector table is obtained by:

- a) A direct conversion of STL powered-flight data from B-GE-to-inertial coordinates. This provides a one-to-one correspondence of supplied-to-calculated vectors. The table usually provides points for several seconds after sustainer engine cutoff (SECO).
- b) Numerical integration by Cowles method, or Runge-Kutta method, using N0CPNI, the same integration program used in the Mercury Program System.
- c) Interpolating for an initial rv (position/velocity vector) by referring to a previously calculated table of vectors (then in memory) using a given time. The velocity vector obtained is then incremented by the velocity of the posigrade rockets; the position vector is left unchanged. With this new rv corresponding to the rv after the firing of posigrade rockets, a new table of vectors is calculated using numerical integration.
- d) Obtaining an initial rv and using the retrofire program R6ATAO to calculate an rv following the retrofire period. This rv is used as the origin of a new table of vectors obtained by numerical integration. In addition, a set of position/velocity vectors is obtained by linear interpolation corresponding to the period during which the retrorockets are fired.

The following steps are taken for each of the stations:

- a) The corresponding station characteristics block is read in from tape.
- b) The distance between each point in the table and the station is calculated until the first point is reached where the distance is less than the indicated value for the station's maximum range.

- c) By linear interpolation between the two consecutive table entries yielding distances greater than and less than the station's maximum range, a starting time is obtained which may be rounded to a multiple of .1 sec. 1 sec. or 6 sec.
- d) Proceeding from the starting time, and using time increments based on the radar (usually six seconds or .1 second), position vectors are interpolated from the table entries by using six-point Lagrangian interpolation. The range, azimuth, and elevation to the station are calculated for each of the calculated vectors. When the calculated range again exceeds the station's maximum range, the Observer program calculates only the ranges to the points in the table. (Observer no longer interpolates for extra values based on the smaller time increment of the radar observations.) When the range is again less than the station's maximum range, the process is repeated. This process is continued until the last value in the table of vectors is tested.
- e) At this point a new station is used and the entire process is repeated.

2.1.4 Usage

In its present state, the Observer program is self-contained except for its part in the SHARE Operating System (SOS).

Parameter Cards—Format and Quantities (all have P in column one)

*	CARD	COLUMN	QUANTITY	FORMAT	UNIT
*	1	3-17	Pitch Angle	SX.XXXXXXXX,SXX	
*	1	19-33	Cant Angle	SX.XXXXXXXX,SXX	DEG.
*	1	35-49	Thrust of Retros	SX, XXXXXXXX, SXX	LBS.
*	1	51-65	Greenwich Hour Angle	SX, XXXXXXXX, SXX	RAD.
*	1	72-72	Card Number (1)		
*	2	3-17	Orbit Weight	SX, XXXXXXXX, SXX	
*	2	19-33	Reentry Weight	SX. XXXXXXXX, SXX	
*	2	35-49	Retrograde Weight	SX. XXXXXXXX, SXX	
*	2	51-65	Retro Burn-out Weight	SX, XXXXXXXX, SXX	LBS.
*	2	72-72	Card Number (2)		
*	3	3-17	Post Escape Rocket Weight		LBS.
*	3	19-33	Spacecraft Area	SX.XXXXXXXX,SXX	SQFT
*	3	35-49	Roll Angle	SX.XXXXXXXX,SXX	DEG.
*	3	51-65	Yaw Angle	SX.XXXXXXXX,SXX	DEG.
*	3	72-72	Card Number (3)		
*	4	3-17	K. Mute, Density Mutila- tion Coff.	SX. XXXXXXXX, SXX	NONE
*	4	72-72	Card Number (4)		

IDENTIFICATION CARD

*	1-2	ID	XX HOL.
---	-----	----	---------

3-3 Blank

3-3 4-72 Any Desired HOL. Informa-

tion

CARD FORMAT FOR STL INPUT

1	1-2	ID	XX HOL.
	4-5	STL Tape File Required	XX INT.
	7-21	Time of Launch (GMT)	SX. XXXXXXXX, SXX MIN.
	41 –41	RV Table Wanted -1	X INT.
	43-43	Output GMT - 0. Elansed - 1	X INT.

CARD FORMAT FOR RV INPUT

*	1	1-2	RV	XX HOL.
*	1	4- 18	Time of Launch (GMT)	SX, XXXXXXXX, SXX MIN.
*	1	20-34	X — Vector	SX, XXXXXXXX, SXX M.U.
*	1	36-50	Y - Vector	SX.XXXXXXXX,SXX M.U.
*	1	52-66	Z - Vector	SX, XXXXXXXX, SXX M.U.
*	1	68-68	RV Table Wanted - 1	X INT.
*	1	70-70	Output GMT - 0, Elapsed - 1	X INT.
*	1	72-72	World Map Wanted - 1	X INT.
*	2	1-15	VX - Vector	SX, XXXXXXXX, SXX M.U.
*	2	17 - 31	VY - Vector	SX. XXXXXXXX, SXX M.U.
*	2	33 - 47	VZ - Vector	SX, XXXXXXXX, SXX M.U.
*	2	49-53	Anchor Time GMT	XXXXX *
*	2	55-56	Integration Interval	XX *
*	2	58-58	* M/S Code 1-Sec., 0-Min.	X INT.
*	2	60-63	Back Integration	XXXX *
*	2	65-6 8	Forward Integration	XXXX *
*	2	70-70	Drag Not Used - 1, Used - 0	X INT.
*	2	72 - 72	Runge-Kutta Used - 0,	
			Cowell - 1	X INT.

EXTEND RETRO CARD

*	1	1-2	XR	XX HOL.
*		4-4	Drag Used - 0, Not Used - 1	X INT.
*		6-6	No. of Rockets Fired	X INT.
*			1 - No. One	
*			2 - No. Two	
*			4 - No. Three	
*			3 - No.S One and Two	
*			5 - No.S One and Three	
*			6 - No.S Two and Three	
*			7 - No.S One, Two and Three	
*		8-8		W TN/T
•			* M/S Code 1 - Sec., 0 - Min.	X INT.
*		10-24	TTF Retro No. One, Elapsed	SX. XXXXXXXX, SXX SEC
*		26-27	Integration Interval	XX *
*		29-32	Forward Integration	XXXX *
*		39-39	B - By Pass Interp., Blank	
			Interpolate	X HOL.
*		41-41	RV Table Wanted - 1	X INT.
*		43-43	Output GMT - 0, Elapsed - 1	X INT.
*		45-59	Long. of Impact	SX, XXXXXXXX, SXX RAD.
*		61-61	World Map Wanted - 1	X INT.
*		72-72	Runge-Kutta Used - 0,	
		·- •-	Cowell - 1	X INT.
			OOMCII - I	A IIII.

EXTEND POSIGR. CARD

1	1-2	XP
	4-6	Drag Used - 0, Not Used - 1
	6-6	No. of Rockets Fired 1, 2 or 3
	8-8	* M/S Code 1 - Sec., 0 - Min
	10-24	TTF Rockets, Elapsed
	26-27	Integration Interval
	29-32	Forward Integration
	39-39	B - By Pass Interp., Blank
		Interpolate
	41-41	RV Table Wanted - 1
	43-43	Output GMT - 0, Elapsed - 1
	51-61	World Map Wanted - 1
	72-72	Runge-Kutta Used - 0,
		Cowell - 1

STATION REQUEST CARD

*	CARD	COLUMN	QUANTITY	FORMAT
*	I	1-1	S Card Code	X HOL.
*		3-4	Desired Station, No. of	XX INT.
*		6-20	Max. Range of Radar at Sta.	SX, XXXXXXXX, SXX SEC.
*		22-36	No. of Seconds Betw. Obser-	
			vations	SX. XXXXXXXX, SXX SEC.
*		38-38	O - No Rounding of Interpo-	
			lated Time,	X INT.
*			1 - Round to Nearest .1 SEC	
*			2 - Round to 1 Second	
*			3 - Round to 6 Seconds	
*		40-40	Output GMT - 0, Elapsed - 1	X INT.
*		42-42	On Last S Card Only,	
			0 - No Other	
*			S Cards Are Expected, 1 -	
			Add. S	
*			Cards May be Used	X INT.
*	I =	1,33	- -	
*	ENI	CARD	Col. 1-3 END	

TYPICAL SET UP FOR DATA CARDS

```
P -3.40000000,+01 +1.35000000,+01 +1.08260000,+03 +0.00000000,+00 1 MA-
P +3.03000000, +03 +2.65294000, +03 +2.79258000, +03 +2.94449000, +03 2 MA-
P + 3.03600000, +03 + 3.02700000, +01 +0.000000000, +00 +0.000000000, +00 3 MA-
                                                                        4 3XD
P + 3.00000000, +01
ID MA9
         3 X DRAG
RV + 0.00000000, +00 + 3.44078309, -01 - 8.07763621, -01 + 5.27785236, -01 1 1 1 MA9
+8.62429042, -01 +4.66370198, -01 +1.52504481, -01
                                                       6 1 0
                                                                   3 2043 0 1 MA9
S 01 + 3.00657150, -01 + 6.00000000, +00 3 1
S02 + 3.00657150, -01 + 6.00000000, +0031
S03 + 3.00657150, -01 + 6.00000000, +00 31
S04 + 3.00657150, -01 + 6.00000000, +0031
S 05 +3.00657150, -01 +6.00000000, +00 3 1
S 06 +3.00657150, -01 +6.00000000.+00 3 1
S 07 +3.00657150, -01 +6.00000000, +00 3 1
S 08 + 3.00657150, -01 + 6.00000000, +00 3 1
S 09 + 3.00657150, -01 + 6.00000000, +00 3 1
S 10 +3.00657150, -01 +6.00000000, +00 3 1
S 11 + 3.00657150, -01 + 6.00000000, +00 3 1
S 12 +3.00657150, -01 +6.00000000, +00 3 1
S 13 + 3.00657150, -01 + 6.00000000, +00 3 1
S 14 + 3.00657150, -01 + 6.00000000, +00 3 1
S15 + 3.00657150, -01 + 6.00000000, +00 31
S 16 + 3.00657150, -01 + 6.00000000, +00 3 1
```

```
S 17 +3.00657150, -01 +6.00000000, +00 3 1

S 18 +3.00657150, -01 +6.00000000, +00 3 1

S 19 +3.00657150, -01 +6.00000000, +00 3 1

S 20 +3.00657150, -01 +6.00000000, +00 3 1

S 21 +3.00657150, -01 +6.00000000, +00 3 1

S 22 +3.00657150, -01 +6.00000000, +00 3 1

S 23 +3.00657150, -01 +6.00000000, +00 3 1

S 24 +3.00657150, -01 +6.00000000, +00 3 1

ID MA-9 3X Drag Reentry

XR 7 1 +1.22362000, +05 4 8888 1 1 1 +3.20355181, +00 1 MA-END
```

OPERATING INSTRUCTIONS

```
TAPES
 A1
        SOS Sys. Tape
 A2
        Blank (SOS)
 A3
        Job Tape
 A8
        BCD Output - RV Table, World Map, Observations
        Print Under Program Control
 B1
        Blank (SOS)
 B2
        Blank (SOS)
 B5
        Blank Intermediate
 B6
        Blank Binary Output, Observations
        Format per 8 Word Record
        Word 1 - Address, Internal Sta. Number
        Decrement, Set Number
        Words 2 and 3
                  - Time Asso. With Observations. BCD Seconds
        Word 4
                 - GMT for Observation. Fixed Binary Seconds
        Word 5
                 - Elapsed Time for Observation. Floating Binary
        Word 6
                  - Slant Range. Floating Point Binary
        Word 7
                  - Azimuth
                                Floating Point Binary
        Word 8
                  - Elevation.
                                Floating Point Binary
 B7
        Station Characteristics Tape Input
 Sense Switches
 SSW No. 4 Down on-line Print of A8
```

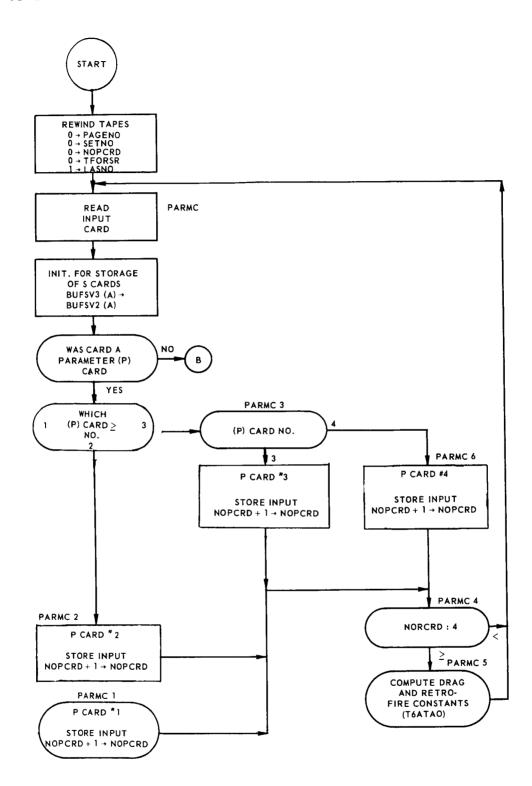


FIGURE 2-1. OBSERVER PROGRAM FLOW CHART (Sheet 1 of 7)

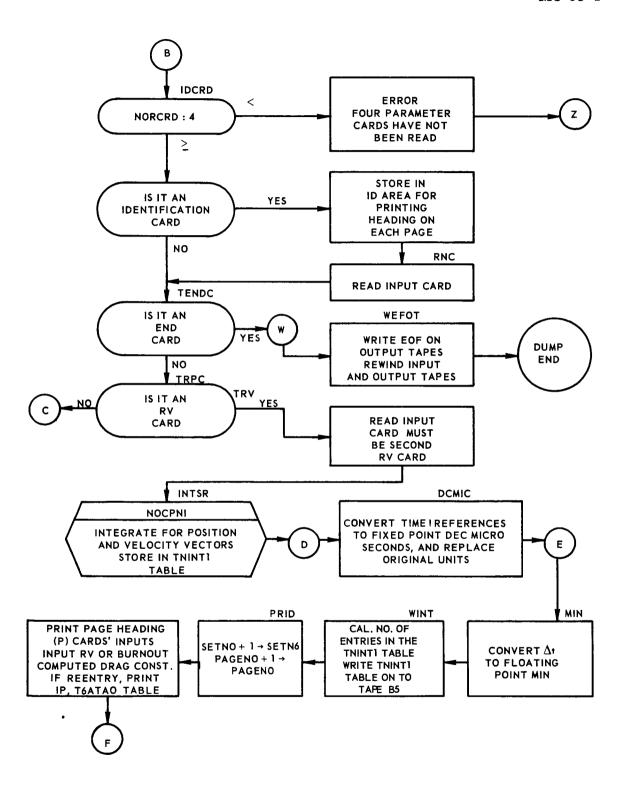


FIGURE 2-1. OBSERVER PROGRAM FLOW CHART (Sheet 2 of 7)

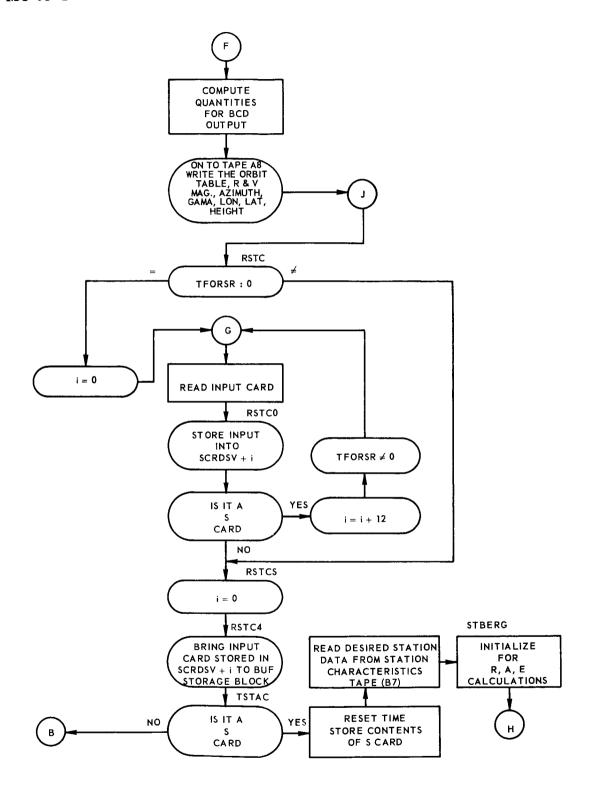


FIGURE 2-1. OBSERVER PROGRAM FLOW CHART (Sheet 3 of 7)

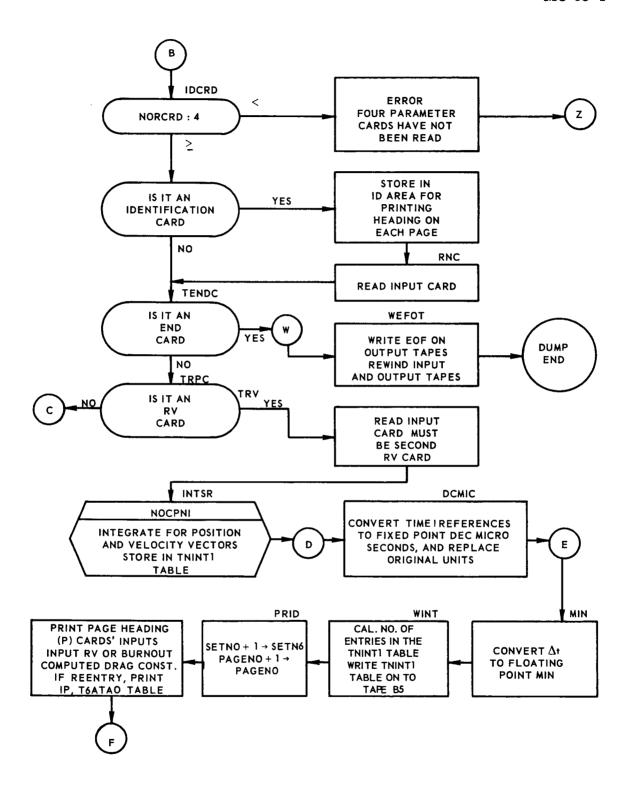


FIGURE 2-1. OBSERVER PROGRAM FLOW CHART (Sheet 2 of 7)

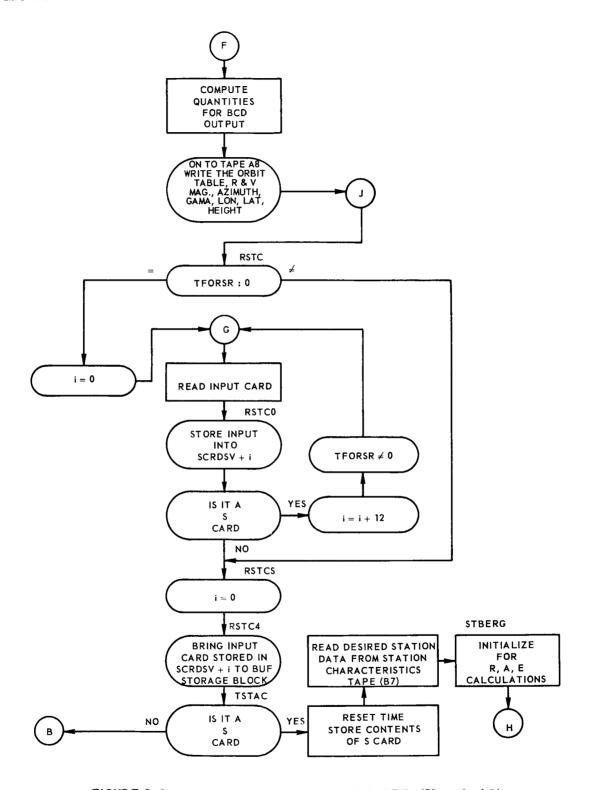


FIGURE 2-1. OBSERVER PROGRAM FLOW CHART (Sheet 3 of 7)

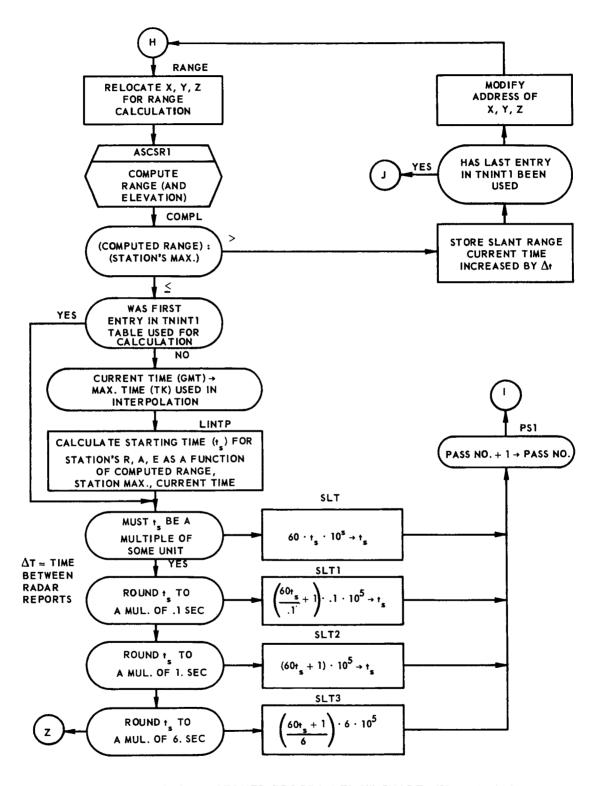


FIGURE 2-1. OBSERVER PROGRAM FLOW CHART (Sheet 4 of 7)

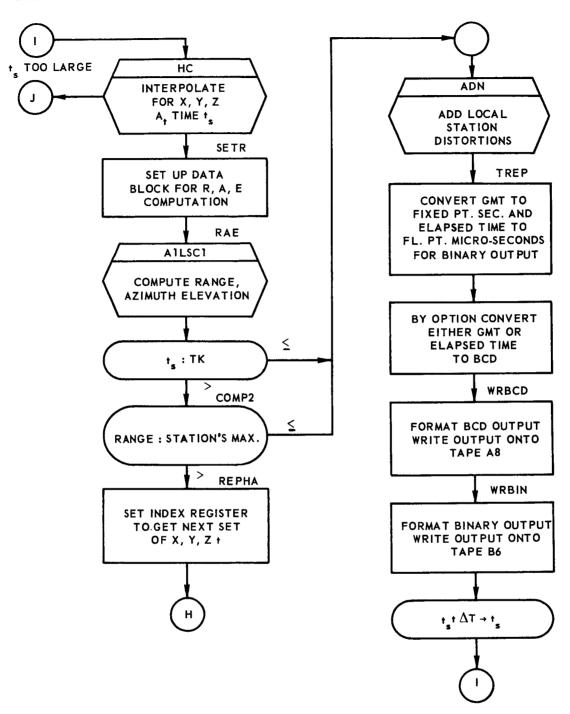


FIGURE 2-1. OBSERVER PROGRAM FLOW CHART (Sheet 5 of 7)

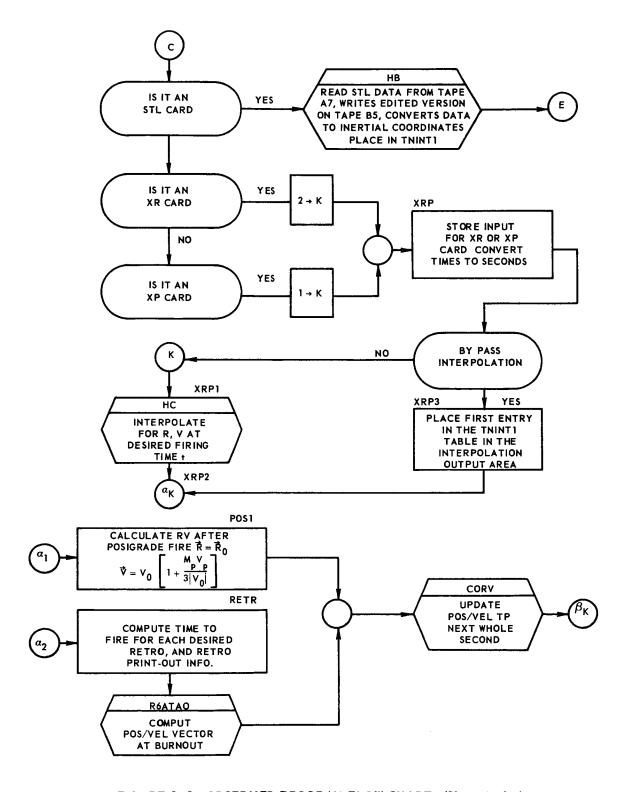


FIGURE 2-1. OBSERVER PROGRAM FLOW CHART (Sheet 6 of 7)

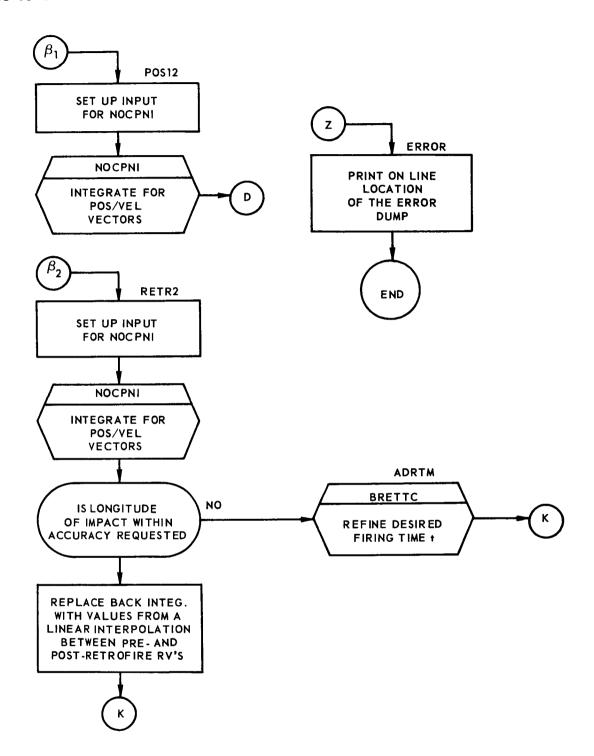


FIGURE 2-1. OBSERVER PROGRAM FLOW CHART (Sheet 7 of 7)

2.2 HB SUBROUTINE

HB assumes the STL launch data to be on a binary tape in a variable number of 24-word records, followed by an end-of-file. The program converts launch (powered flight) data from B-GE coordinates in fixed-point STL units to inertial coordinates in floating-point Mercury units. The output is placed in the integration table (TNINTI). An average mean (Δt) is computed and used as the base value for this integration table.

The flow chart for HB is shown in Figure 2-2.

All records from the binary input tape are read and a check is made of the 12-bit folded checksum. The folded checksum constitutes part of the first word of each record. Since each item of STL data is a 24-bit word, the 12 bits in positions 24 through 35 are always zero. Items 2 through 10 are the values HB uses.

2.2.1 Input Format

- 1. Logical word from column binary card
- 2. Discreet quantities
- 3. Č
- 4. $\dot{\eta}$
- 5.
- 6. T_E
- 7. °C
- 8. η
- 9. *E*
- 10. Special B-GE checksum
- 11-24. Not used by HB

2.2.2 Output Format

TNINT1	+ ∞		
+ 1	Δt	+ 7	x
+ 2	To	+ 8	Ý
+ 3	$\mathbf{T_n}$	+ 9	Ż
+ 4	X	+ 10	X
+ 5	Y	•	•
+ 6	${f z}$	•	•

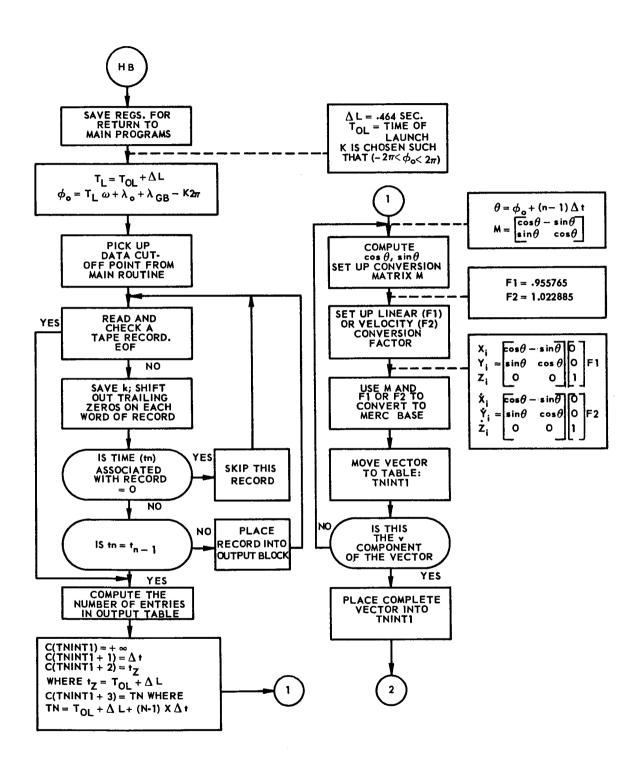


FIGURE 2-2. HB FLOW CHART (Sheet 1 of 2)

FIGURE 2-2. HB FLOW CHART (Sheet 2 of 2)

2.3 HC SUBROUTINE

HC performs a 6-point Lagrangian interpolation. It consists mainly of a control program around the SHARE routine INTP2. If a 6-point Lagrangian interpolation cannot be performed, a simple linear interpolation is executed.

The flow chart for HC is shown in Figure 2-3.

The input to the subroutine is the time of the desired position and velocity vectors; output is the vectors corresponding to the time requested.

2.4 RAE SUBROUTINE

This subroutine applies errors to R, A, E's and prepares a profile tape and a listing tape. The flow chart for RAE is shown in Figure 2-4.

2.5 RFBRAE SUBROUTINE

RFBRAE is an output routine. Its major purpose is to operate with station characteristics information. The operation data consists of range, azimuth and elevation. Azimuth and elevation data is converted to degrees; range data is converted to yards. An option is incorporated in this routine which allows printing this information on-line or on the listing tape. Errors can be introduced in this routine due to local vertical, refraction and boresight corrections by requesting specific values from the station. This interrogation is accomplished in the station characteristic block.

The flow chart for RFBRAE is shown in Figure 2-5.

Notes: This information pertains to Figure 2-5.

 $K = 1.161466225 \times 10^{-3}$

Ng: Modulus (N-1)-27th word of station characteristics block.

Δλ: Local vertical deflection—fifteenth word

 $\Delta \phi$: Local vertical deflection—sixteenth word

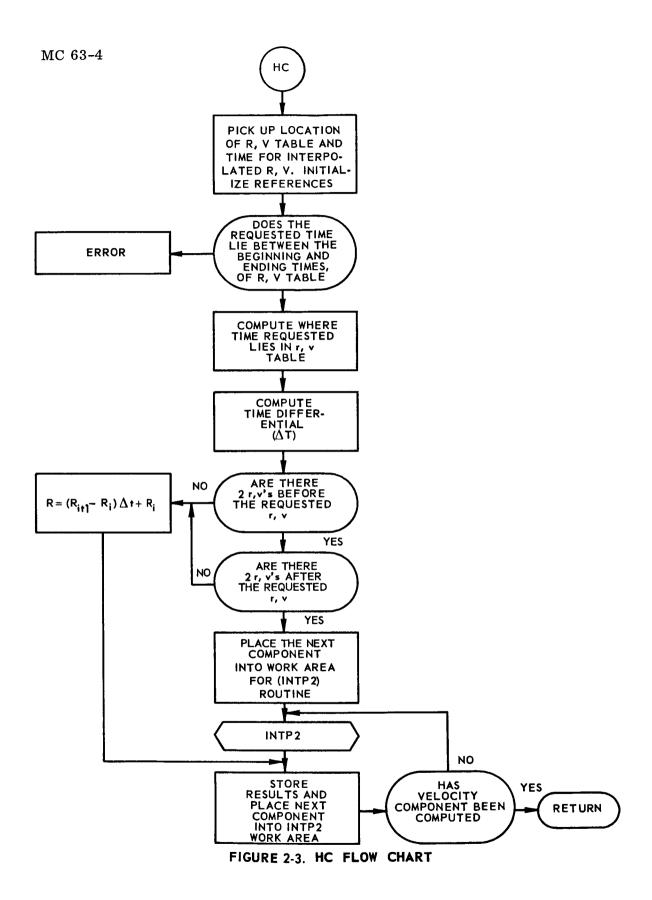
C2: Boresight azimuth correction—thirty-third word

C₃: Boresight elevation correction—thirty-second word

D₂: Inertial longitude at reference time—nineteenth word

2.6 RNRCRD SUBROUTINE

The RNRCRD subroutine reads the integration tape into storage. The flow chart for RNRCRD is shown in Figure 2-6.



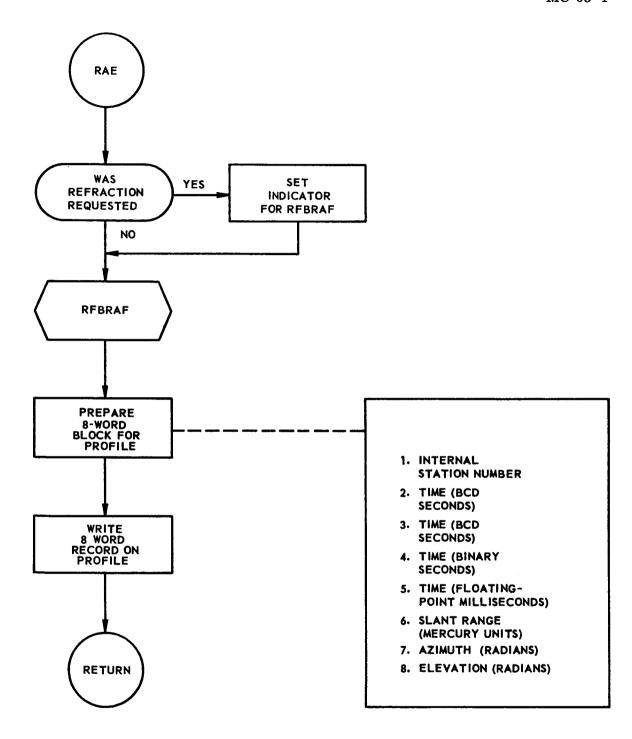


FIGURE 2-4. RAE FLOW CHART

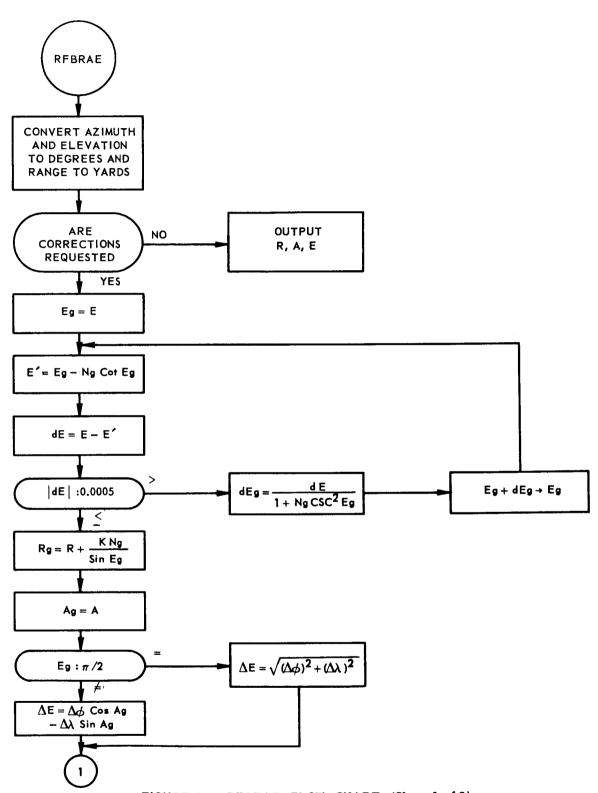


FIGURE 2-5. RFBRAE FLOW CHART (Sheet 1 of 2)

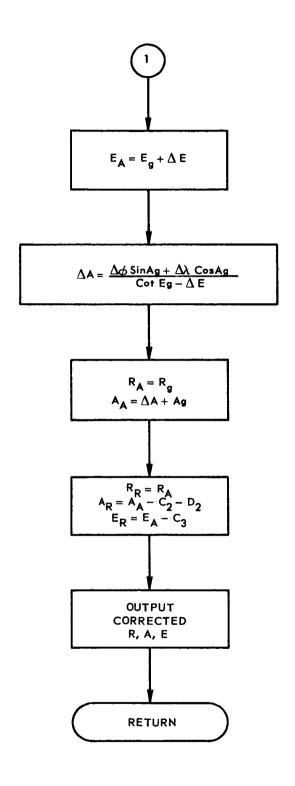
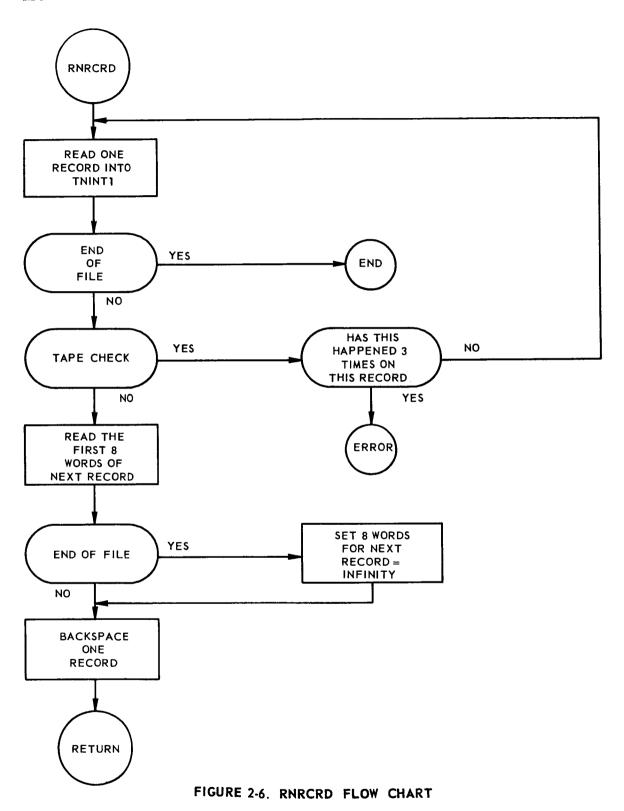


FIGURE 2-5. RFBRAE FLOW CHART (Sheet 2 of 2)



2-26

2.7 RVCAL SUBROUTINE

This subroutine calculates position and velocity quantities for the simulated trajectory. Input to the routine is a table of position and velocity vectors, and the output consists of tables of values on a listing tape. During the orbit phase, one table of orbital elements is also provided.

The equations used in obtaining the trajectory quantities and orbital elements appear on the following pages.

The trajectory table includes values for:

The six vector components $(x, y, z, \dot{x}, \dot{y}, \dot{z})$

Range magnitude (R)

Inertial velocity (V_T)

Relative velocity (V_p)

Inertial gamma (γ_{T})

Relative gamma (γ_R)

Inertial azimuth (A_T)

Relative azimuth (A_R)

Geocentric latitude

Geodetic latitude

Relative longitude

Inertial longitude

Altitude above an oblate or spherical earth.

The orbital element table includes:

Semimajor axis (a)

Mean motion (n)

Period (T)

Eccentricity (e)

Eccentric anomaly (E)

Mean anomaly (M)

Inclination angle (I)

Argument of the ascending node (Ω)

Argument of perigee (ω)

Longitude of perigee (L)

True anomaly (TA)

Apogee (A)

Perigee (P)

Constants which are used in the equations:

 R_{ρ} (radius of the earth) = .999251039 Mercury units

 V_{e} (rotational velocity of earth) = .058833543 Mercury time units

Trajectory Calculations

$$\begin{split} R &= \sqrt{\dot{x}^2 + y^2 + z^2} \\ V_I &= \sqrt{\dot{x}^2 + \dot{y}^2 + \dot{z}^2} \\ x_R &= \dot{x} + y \omega_e, \ \dot{y}_R = \dot{y} - x \omega_e, \ \dot{z}_R = \dot{z} \\ V_R &= \sqrt{\left(\dot{x}_R\right)^2 + \left(\dot{y}_R\right)^2 + \left(\dot{z}_R\right)^2} \\ \gamma_I &= \sin^{-1} \frac{x \dot{x} + y \dot{y} + z \dot{z}}{R \ V_I} \\ \gamma_R &= \sin^{-1} \frac{r \cdot V_R}{R \ V_R} \end{split}$$

$$A_{I} = \cos^{-1} \frac{R\dot{z} - z V_{I} \sin \gamma_{I}}{V_{I} \cos \gamma_{I} \sqrt{x^{2} + y^{2}}}$$

$$A_{R} = \cos^{-1} \frac{R\dot{z}_{R} - z V_{R} \sin \gamma R}{V_{R} \cos \gamma_{R} \sqrt{x^{2} + y^{2}}}$$

Latitudes, longitudes and heights are obtained by use of A3MSCP (see MC 63-3).

Orbital Calculations

$$e \cos E = R (V_I)^2 - 1$$

$$a = \frac{R}{1 - e \cos E}$$

$$n = a^{-\frac{3}{2}}$$

$$T = \frac{2\pi}{n}$$

e Sin E =
$$\frac{\vec{r} \cdot \vec{v}}{\sqrt{a}}$$

$$e = \sqrt{(e \sin E)^2 + (e \cos E)^2}$$

$$E = \tan^{-1} \frac{e \sin E}{e \cos E}$$

$$M = E - e \sin E$$

$$\sqrt{p} = |\vec{r} \times \vec{v}|$$

$$R_{x}^{*} = \frac{(\overrightarrow{r} \times \overrightarrow{v}) \cdot \overrightarrow{i}}{\sqrt{p}}, R_{y}^{*} = \frac{(\overrightarrow{r} \times \overrightarrow{v}) \cdot \overrightarrow{j}}{\sqrt{p}}, R_{Z}^{*} = \frac{(\overrightarrow{r} \times \overrightarrow{v}) \cdot \overrightarrow{k}}{\sqrt{p}}$$

$$\begin{aligned} &\operatorname{Cos} \ \mathbf{i} = R_{\mathbf{X}}^{*} \\ &\operatorname{Sin} \ \mathbf{i} = \left(R_{\mathbf{X}}^{*2} + R_{\mathbf{y}}^{*2} \right)^{1/2} \\ &\operatorname{I} = \tan^{-1} \left(\frac{\sin \mathbf{i}}{\cos \mathbf{i}} \right) \\ &\operatorname{Sin} \ \Omega = \frac{R_{\mathbf{X}}^{*}}{\sin \mathbf{i}}, \ \operatorname{cos} \ \Omega = -\frac{R_{\mathbf{y}}^{*}}{\sin \mathbf{i}} \\ &\Omega = \tan^{-1} \left(\frac{\sin \Omega}{\cos \Omega} \right) \\ &\overline{P} = \frac{\cos E}{R} \ \overrightarrow{r} + \sqrt{a} \sin E \ \overrightarrow{v} \end{aligned}$$

$$\overline{Q} = \frac{\sin E}{R(1 - e^{2})} \frac{1}{2} \ \overrightarrow{r} + \frac{\left(\sqrt{a} \left(\cos E - e\right)}{(1 - e^{2})^{1/2}} \ \overrightarrow{v} \end{aligned}$$

$$\operatorname{Cos} \ \omega = \operatorname{Cos} \ \Omega \operatorname{Px} + \operatorname{Sin} \ \Omega \operatorname{Py}$$

$$\operatorname{Sin} \ \omega = -\operatorname{Cos} \ \Omega \operatorname{Qx} - \operatorname{Sin} \ \Omega \operatorname{Qy}$$

$$\omega = \tan^{-1} \left(\frac{\sin \omega}{\cos \omega} \right)$$

$$L = \Omega + \omega$$

$$T_{\mathbf{A}} = \tan^{-1} \left(\frac{\overrightarrow{r} \cdot \overrightarrow{Q}}{\overrightarrow{r} \cdot \overrightarrow{P}} \right)$$

$$A = \left(\frac{1 + e}{a} \right) - R_{e}$$

 $P = \left(\frac{1 - e}{a}\right) - R_e$

2.8 SELECTOR PROGRAM

The Selector program selects a flight profile for the spacecraft from a given set of radar observations and applies the following parameters to the radar reports:

- a) Begin-transmission mark
- b) End-of-transmission mark
- c) Valid or invalid transmission identification
- d) Random error code
- e) Pathological error code
- f) Transmission error code
- g) Transmission delay
- h) Bias error code

The flow chart for the Selector program is shown in Figure 2-7.

2.8.1 Input Requirements

The following programs are used with the Selector program: RCDI, GLFILE, DFLN, and CSTI.

A binary input tape (output from Observer program) is also required and its format is as follows:

- Word 1—address portion of the word contains the internal station number of the radar installation.
- Word 2 and Word 3—time of observation associated with radar readings. The time is the total number of seconds in BCD.
- Word 4—total number of seconds since midnight preceding launch, in floating-point binary
- Word 5—total number of milliseconds since launch, in floating-point binary
- Word 6-slant range, in floating-point binary

Word 7—azimuth, in floating-point binary

Word 8—elevation, in floating-point binary

Input cards (called station request cards) are required. These station cards control the output and various identifications, tags, and delays which are applied to the radar reports from the requested station. At least one input card is required per station requested; however, there is no limitation on the number of cards in the station request deck. The last card of the station request deck must be an END card (END punched in columns 1-3). The station request card format is as follows (unless otherwise specified, all leading zeroes are to be punched):

C	oŀ	umns	•

<u> </u>			
1-2	The subchannel of the DCC to be used.		
3-4	The internal station number.		
7-15	The BCD time associated with the first observation desired. The columns are to be punched as follows:		
	xxxxxx.xx Any leading zeroes are left blank.		
17-25	The BCD time associated with the last observation desired. Punch columns the same as shown above for columns 7-15.		
27	M code: the m code is used to tag the report with a begin-transmission or end-of-transmission mark and a valid or invalid-data identification.		
29-30	Transmission error code.		
32-33	Pathological code.		
35-36	Random error code.		
	(NOTE: The random, transmission, pathological and bias error codes explained in the Shred program write-up.)		
38-52	The transmission delay desired for this station, in floating-point milliseconds.		
	1 v vvvvvvv +VV		

Columns:

Type of radar: punch H for high-speed or leave blank if

low-speed.

67-68 Set number associated with the station requested.

70-71 Bias error code.

The station request cards must be in sequence by set number.

2.8.2 Output Requirements

A binary output tape is generated and is used as input to the Shred program. Each tape recorded contains 800 words, eight words per logical record. The logical record format is as follows:

Word 1—the decrement contains the transmission error code.

Bits 23 to 26 contain the pathological error code; bits 28 to 32 contain the random error code; bit 33 represents the valid or invalid identification (bit represents valid data); bit 34 represents the start-of-transmission tag; and bit 35 represents the end-of-transmission tag (bit indicates presence on the latter two tags). Bits 8 to 11 contain the bias error code.

Word 2—time of receipt of the first character by the computer, in milliseconds, binary integer.

Word 3-subchannel of DCC, binary integer

Word 4—time of observations in BCD:

xx Hrs xx Min xx Secs

Word 5—internal station number, binary integer

Word 6—slant range, in floating-point yards

Word 7—azimuth, in floating-point degrees

Word 8-elevation, in floating-point degrees

High padding is used in the last tape record.

2.8.3 Method

The noise and errors injected into the data fall into four classes; numerous variations are possible within each class. The four classes are:

- a) Random errors—random noise generated by the radar set in making an observation.
- b) Transmission errors—the random noise introduced by the transmission system.
- c) Pathological errors—the nonrandom failures which creep into the system to cause dropping or garbling of bits, words, or transmissions due to outright failures in the system. Any conceivable trouble can be introduced in this class.
- d) Bias errors—the consistant algebraic bias of data caused by misalign—ment of tracking equipment.

The order of application of the classes of errors is bias, random, pathological, and transmission. This sequence conforms most closely to their order of actual occurrence. If both bias and random errors are applied, the random errors are applied to the data which has been previously biased.

Errors of all four classes are injected in the data in any combination of available variations, thus offering the facility of changing the variations from one section of the data to another. This changing of variations is accomplished by sectioning the data by means of the time interval cards which serve as input to the Selector program. Each group of radar site observations can be divided into arbitrary time intervals, and within each interval one variation of each class of errors may be applied.

In general, several time interval cards would be prepared for each site. The site code and the teletype channel over which the radar transmissions are to be sent are given on this card. The time interval corresponding to that card is specified by t_i and t_f . The variation of each class of errors to be applied to that interval is specified by error codes. Also specified on this card is a transmission delay associated with this section of data; this information, in conjunction with the time of observation, determines when the data arrives at the input to the computer. A further item given on this card is information regarding teletype control signals needed to complete the generation of a transmission.

Using the time interval cards and the tape generated by Observer as inputs, Selector generates a tape containing the selected radar data and associated system error codes in proper form from input to the next data generation program, the Shred program.

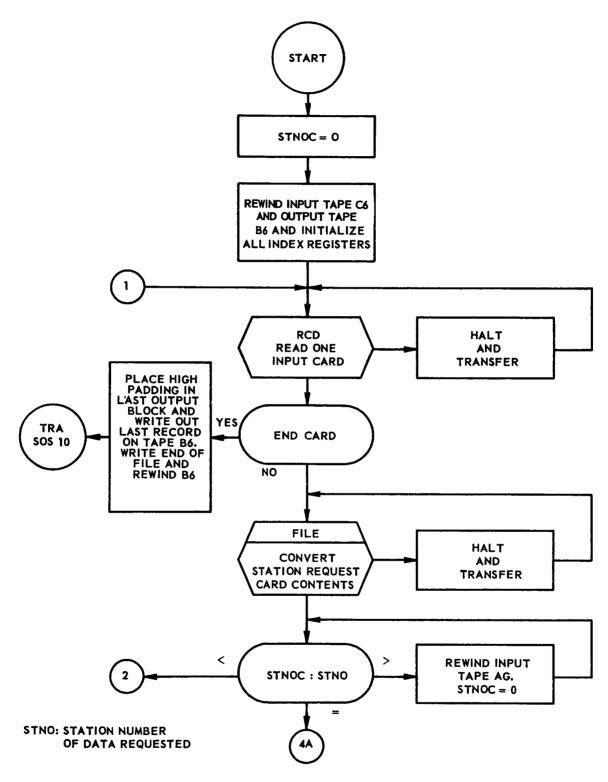


FIGURE 2-7. SELECTOR PROGRAM FLOW CHART (Sheet 1 of 5)

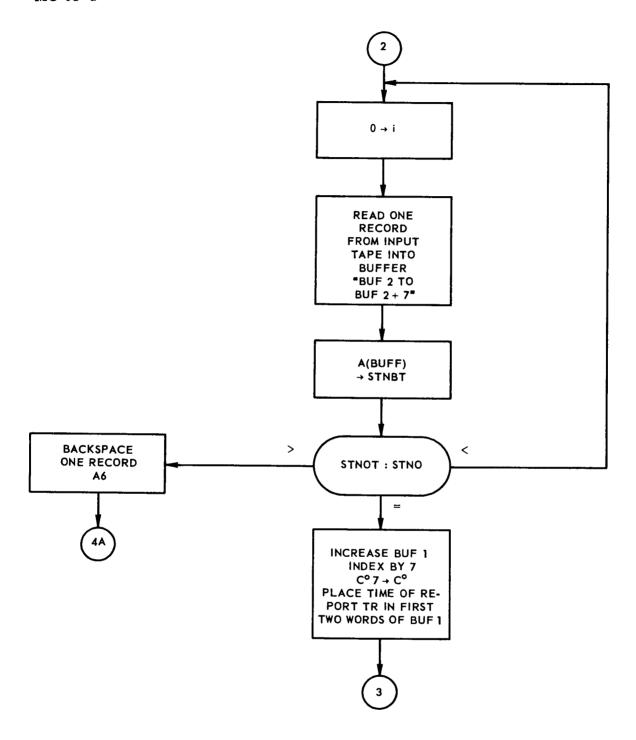


FIGURE 2-7. SELECTOR PROGRAM FLOW CHART (Sheet 2 of 5)

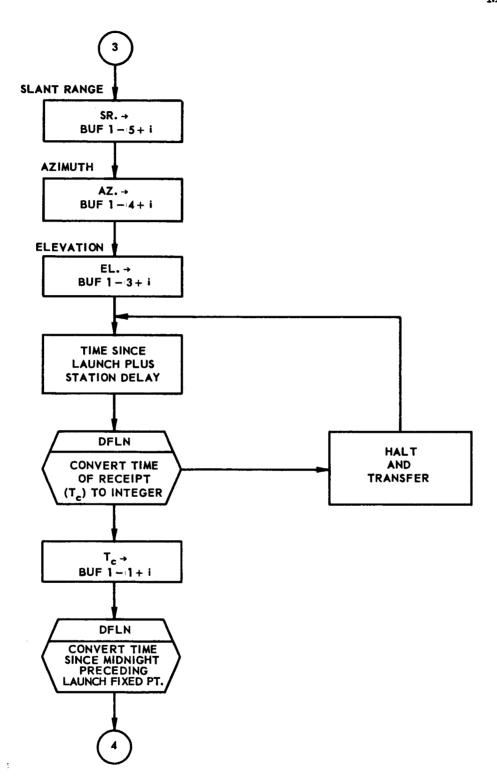


FIGURE 2-7. SELECTOR PROGRAM FLOW CHART (Sheet 3 of 5)

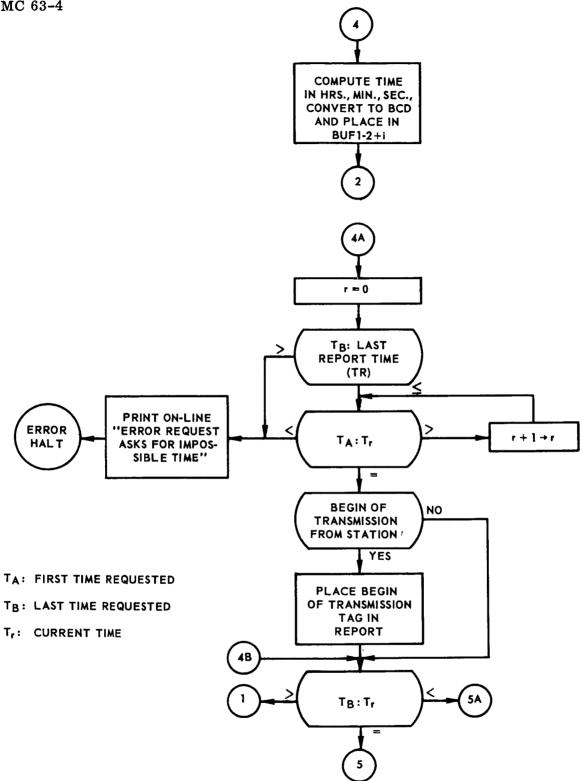


FIGURE 2-7. SELECTOR PROGRAM FLOW CHART (Sheet 4 of 5)

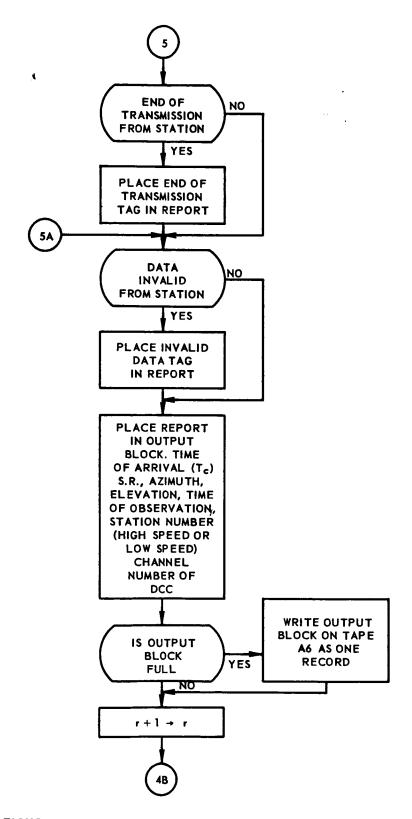


FIGURE 2-7. SELECTOR PROGRAM FLOW CHART (Sheet 5 of 5)

2.9 SHRED PROGRAM

The Shred program produces simulated Mercury mission data in real time for the Goddard computer.

A schematic diagram of Shred is shown on Figure 2-8.

Shred produces the following simulated input for SIC:

- a) IP 7094 data
- b) B-GE data
- c) Telemetry readings
- d) Low-speed TTY readings
- e) High-speed radar readings

for Goddard

2.9.1 Input Requirements

There are two input tapes used with the Shred program: (1) the launch tape provides simulated B-GE and IP 7094 launch data, and (2) the sequence of radar readings produced by the Observer program and selected by the Selector program.

Shred is controlled by the radar input. Each logical record of radar information (or observation) contains:

- a) Type of output required
- b) Time at which the message is to be simulated upon arriving at the computer
- c) Perturbation codes (PEC, REC, and TEC)
- d) Begin or end-of-transmission codes
- e) Validity code
- f) Range, azimuth, and elevation (RAE) values
- g) Time of RAE observation
- h) Subchannel of DCC to be used
- i) Station identifications

The input to Shred is an 800-word per record tape containing 100 radar readings, each reading consisting of eight words. The 8-word readings are referred to as logical records.

2.9.2 Output Requirements

The output of the Shred program is contained on three tapes:

- a) Low-speed output for Goddard
- b) High-speed output for Goddard
- c) Bermuda high-speed output for Goddard

The low and high-speed output to Goddard must first be independently sorted by time-of-arrival sequence (since the size of the logical record is different for high-speed and low-speed data), then merged to form one SIC input tape.

2.9.3 Method

A radar reading is provided by the observer-selector complex each time a reading is requested (there are some exceptions to this in the launch phase). Each reading contains a PEC, REC, TEC and BEC (pathological, random, transmission and bias error codes). These codes are used to obtain controlled perturbations in the Shred (teletype message format) output message.

2.9.4 Usage

Shred uses a series of tables as its input to provide a flexible output. Shred usage reduces to a description of these tables:

<u>Table</u>	Function of	Provides
T1	internal station number	the subroutine to be used for each particular Shred input reading
Т3	channel number	the Shred region to be used to format messages
T4	station number	the first seven characters of TTY transmitted radar mes- sages for Goddard

<u>Table</u>	Function of	<u>Provides</u>
Т5	station number	conversion factors from Shred input to simulated input
SPTEi	TEC (transmission error code) i=1,2,N	 three probablities for a transmission error and three conditional probabilities which determine the type of transmission error, given by that such an error exists for TTY messages only, there are three sets of the above six probabilities, one set each for the beginning, body and end of transmission of the message
SPCVi	radar type	conversion factors from Shred radar input units to Shred radar output units. Applies biased errors
SPICi	radar set	the first seven characters, in octal TTY code, of each message that will be sent by a particular radar site
RER	REC (random error code)	standard deviations for application as random errors to range readings for all radars. Any error code of zero causes that particular type of error to be bypassed, i.e., no error
REA	REC	standard deviations for application to radar azimuth readings
REE	REC	standard deviations for application to radar elevation readings
PER	PEC (pathological error code)	numbers used to simulate patho- logical errors in radar range readings
PEA	PEC	simulated pathological errors for radar azimuth readings

Table	Function of	Provides
PEE	PEC	simulated pathological errors for radar elevation readings
PET	PEC	simulated pathological errors for observation time in radar readings, applicable to both Goddard low-speed and Bermuda radars
HP1	PEC = 1, 2,15	the pathological errors to be applied to the simulated positionvelocity vectors of the B-GE
HP2	PEC	the pathological errors to be applied to the simulated positionvelocity vectors of the IP 7094
HR1	REC-1,2,15	the standard deviations for ran- domly perturbing the simulated position - velocity vectors of the B-GE
HR2	REC	the standard deviations for ran- domly perturbing the simulated position - velocity vectors of the IP 7094
HST1	telemetry bit	the telemetry schedule for the launch phase
HST1A		manual reverse: telemetry
HST1B	telemetry bit	probabilities that the telemetry schedule is in error also ECT, RFT, GEB line 1 HSRR, IP 7094 line 2
HST2		the periods of time during which simulated B-GE data is to be produced. The source of this data is the STL simulated data. If no STL message exists for a given time, no data will be produced for that specific time. The STL message time is used. Sometimes Shred modifies this time to reflect time delays, etc.

<u>Table</u>

Function of

Provides

HST3

the periods of time during which IP 7094 data is to be produced. The IP 7094 data is interpolated from an error-free set of position-velocity vectors provided by STL. Messages are produced for the entire "on" periods of time, even if these times do not correspond to time of the r-v table. The latter values, however, are meaningless except perhaps for the telemetry

Table HST2 and HST3 are of the form:

HSTi DEC first time data requested

DEC end of first request

DEC ith time data requested

DEC end of ith request

DEC nth time data requested

DEC end of nth request

OCT 37777777777 (large number)

All times are given as binary integers in microseconds, i.e., XX.XXE6B35 where XX.XX is in seconds

HST11

Miscellaneous input parameters:

HST11 BCI I,XXYYZZ

BCI I,0000AA

BCI I, PPQQRR

DEC 1

<u>Table</u>	Function of	Provides
		where: XXYYZZAA is the GMT of launch for Bermuda radars in hours, minutes, seconds and tenths of seconds. PPQQRR is the ECT of launch for telemetry messages in hours, minutes and seconds. γ is the angle from the launch pad to Greenwich (the angle between Mercury inertial and IP 7094X-axes at launch in floating-point radians. One value is 1.40581173. The value varies slightly with different pads)
HST12		the times, in microseconds, at which to change the retrofire setting in the spacecraft. The table is of the form:
		HST12 DEC time for first change
		DEC time for second change
		DEC time for last change
		OCT 37777777777
		This table is used in conjunction with HST13
HST13		the retrofire settings for the spacecraft clock. The table is of the form:
		HST13 BCI 1, original clock setting
		BCI 1, second setting
		:
		BCI 1, last setting

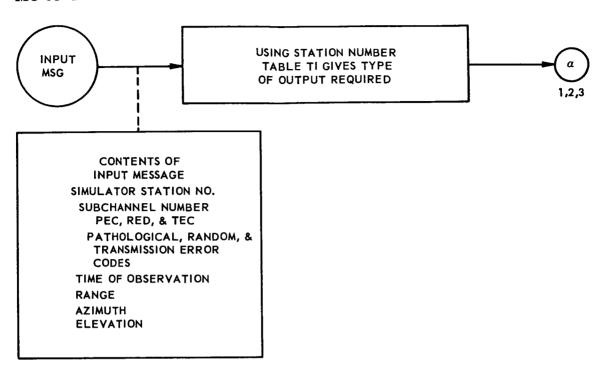
Table	Function of	<u>Provides</u>
		The settings are ECT's since launch of the form XXYYZZ (hours, minutes and seconds). When $HST12+i\le t < HST12+i+1$, where t is message time, the clock setting used is $HST13+i+1$
HST14		the periods of time during which Asuza data is to be produced. The simulated source for this data is the STL IP 7094 data. Shred tags the IP 7094 data with a minus sign to distinguish the Asuza source from the IP 7094 source. The table is of the same form as table HST2
METRY		the telemetry schedule for the launch phase. All times are referenced to launch. (If launch is not at midnight, see HST 11). The times are given in fixed-point milliseconds. The table is of the following form:
		METRY DEC XXX.XXX : : : : : : : : : : : : : : : :
The ordered eve	ents are:	
	METRY + 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 + 11	Start Bermuda solution One of three posigrades fired Two of three posigrades fired Three of three posigrades fired One of three retros fired Two of three retros fired Three of three retros fired Liftoff Escape tower released Escape tower rockets fired Spacecraft separation Abort sequence initiated

+12	Abort phase started
+13	Orbit phase started
+14	SECO

When the time of a message being generated is equal to or greater than the time the telemetry event is scheduled to occur in METRY, all messages thereafter indicate that the event has occurred. Because of this slight lag in the assignment of an event to messages, the times in METRY should precede the actual times by one or two seconds.

SHRED:	ADDITIONAL INPUT DATA	
Symbol	Op Code	Var Field
HSPGOD	EQU	A nonzero value requests that the particular type of data be produced, i.e., high-
LSPGOD	EQU	speed data from Cape Canaveral to Goddard, low-speed TTY data from radar
HSPBRM	EQU)	stations to Goddard and high-speed input for the Bermuda computer. The presence of a nonzero value will cause a tape to be labelled appropriately and will erase any superfluous data in the output regions at the end of the output data.
ADJTM	EQU	A nonzero value produces messages in the new telemetry format. A zero value produces messages in the old TTY format. The new format is designed to lessen non-recognition of poorly transmitted messages by repetition of key information. Is is the sole format in use, except for transitionary testing, at present. Pertains to Goddard Shred only.
THSML	EQU	A nonzero value gives the FPS/16 radar a maximum range of 1000 nautical miles (2 ¹⁰). If zero, the FPS/16 has a range of 500 (2 ⁹) naut. miles. Pertains to Goddard Shred only.

Symbol	Op Code	Var Field
HSGODD	BCI	5,
	BCI	5, There must be a 10-word heading or heading space provided for each sym-
LSGODD	BCI	5, bol of HSPGOD, LSPGOD and HSPBRM that is nonzero. Even
	BCI	t through the heading on the low-speed output tape is lost during variable length merge, reading operations de-
HSBERM	BCI	mand that this tape have a heading. If a heading is omitted, the first record
	BCI	of that tape will be read as the heading and the remaining information on the tape will be misinterpreted.



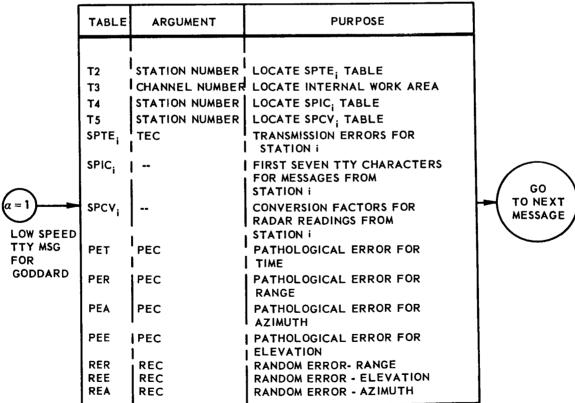


FIGURE 2-8. SCHEMATIC DIAGRAM (SHRED TABLES) (Sheet 1 of 3)

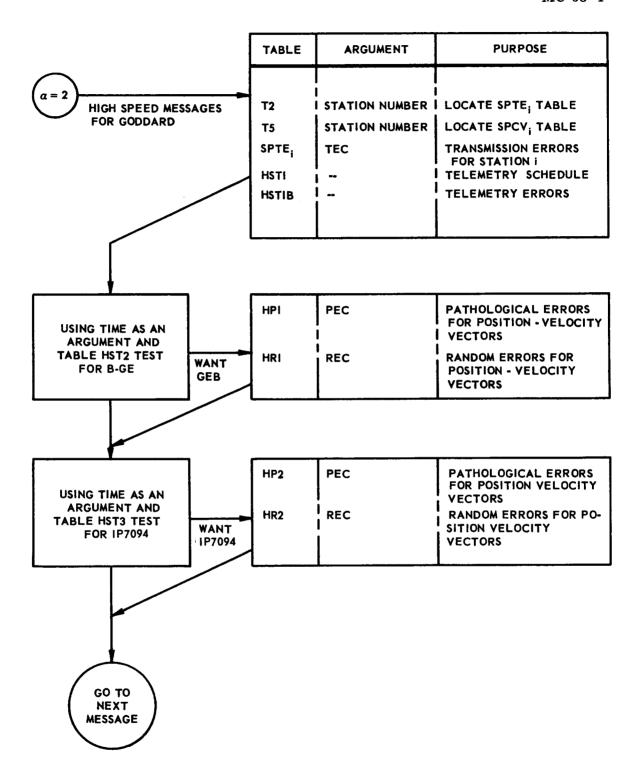


FIGURE 2-8. SCHEMATIC DIAGRAM (SHRED TABLES) (Sheet 2 of 3)

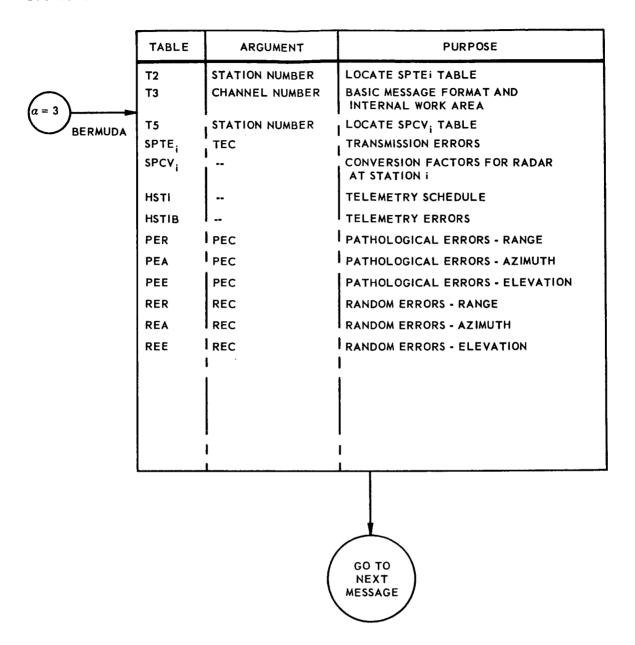


FIGURE 2-8. SCHEMATIC DIAGRAM (SHRED TABLES) (Sheet 3 of 3)

2.10 SORT PROGRAM

The Sort program arranges radar observations recorded on tape according to their time of arrival to the computer. This program sorts either the output of the Selector program for Bermuda tapes or the output of the Shred program for all tapes generated for Goddard reception.

Figure 2-9 shows the flow chart for the Sort program.

2.10.1 Input Requirements

- a) A binary tape containing radar observations.
- b) Alter cards, recognized by SOS, of the following types:
 - 1) An equals card which identifies the size of the tape record of the binary tape.
 - 2) An equals card which identifies the size of the logical record length of data within the tape record.
 - 3) An equals card which identifies the size of the logical record.
 - 4) An equals card which indicates whether the input tape does or does not contain a tape label.
- c) The program also has the ability to sort, by means of an equals card, the input tape set at any logical setting on a given channel.

The Sort program requires four tapes: one is the original input tape; the others are required for successive merge passes in the Sort program. Two of the four tapes are on channel A; the original input tape and the remaining tape are on channel B. Logical settings for the other three tapes are variable and can be set by an equals card as input to SOS.

2.10.2 Output Requirements

The only output requirement for the Sort program is the selection of the output channel for the final tape. The output tape can be obtained on either channel A or B; selection is set by another equals card to the program. The output tape's record length originates from the same record length as does the input.

2.10.3 Method

There are two main phases of operation in the Sort program. The first is the sorting phase, where records on the input tape are sorted according to the key word of the sort operation. The second phase merges the records sorted by time of arrival to the computer and arranges them so the final output tape is sorted by information within each record and also by the records themselves.

In the first phase, the program tests the input tape to determine if it has a label. If the tape has a label, the program reads the label and saves it for future reference before processing begins. Initially, the program reads the first two records from the input tape into two separate buffers. Then, the program sorts the information in one record, according to the key word set by the equals card, and stores the information on one of the tape units on channel A. Sort reads the next tape record into the buffer unit just emplied and continues to the second buffer. Again, the program sorts information in the record but stores it on the opposite tape on channel A. The program continues this operation with the different records on the input tape, continually storing the sorted information in the records alternately on the two tapes of channel A. The first phase is completed when the program reaches an end-of-file indication on the input tape. The Sort program then rewinds the input tape and the two output tapes on channel A.

During the second, or merge, phase the Sort program compares the key words from the first record of the first tape on channel A. This comparison is made to determine which key word is in the lowest-order sequence. Once this sequence is established, the key word and its corresponding logical records are stored in the output buffer. The program continues this operation until the output buffer is filled and a corresponding record is written on the output tapes. Therefore, logical records could come, partially, from each record from each channel A input tape. The procedure used to record records is as follows: Two records are alternately recorded for the first pass on each tape. For example, two sorted records are recorded on tape B1, and the next two on B2, etc., until the pass is complete.

The procedure for the second pass is similar to the first pass except that the records are stored as four records to each tape. The records of each pass are recorded according to powers of two, beginning with two to the first power. The second pass is recorded at two to the second power, or four records to each tape; the third pass is recorded at two to the third power, or eight records to each tape, etc.

The final pass of the merge phase is reached when the Sort program recognizes that it has reached the point where it has written all of its input as output on one tape for that pass. For example, if the input consists of a total of 64 records, the final merge pass presents a powers-of-two configuration that has recorded all 64 records on one tape. Since the information has now

elapsed, records will never be placed on the other tape. This signals the end of the merge pass.

The final step of the Sort operation is to determine if the final output tape is on the program output channel, as indicated by an input parameter to the program (since the channel on which the output will be recorded depends on the number of records of input and how badly they are originally out of sort). If Sort has determined that the output is now on a channel which was not requested by the input parameter, it will essentially go through one more pass of transferring the data from the channel on which it is now to the requested output channel. However, if the tape information is on the requested channel, the program is finished. The program writes on end-of-file message upon the final output tape, rewinds the tape, and transfers to SOS for completion.

Tape output formats are shown in tables 2-1 through 2-4.

2.10.4 Usage

a)	Tape Set-Up:	A1	SOS System tape	Check Tape
		A2	Blank list tape	L.D.
		A3	Job tape	L.D.
		A5	Blank	H.D.
		A7	Blank	H.D.
		B1	Blank	L.D.
		B2	Blank	L.D.
		B3	Blank	L.D.
		B6	Shred input	H.D.

- b) Sense Switch Setting-none.
- c) Key Setting-none.
- d) Operating Instructions—clear machine and load tape.
- e) Halts:

HPR 44444 ₈	Remove B6 and replace with a blank. Press START.
HPR 33333 ₈	Tape read error. Press START to continue; if error persists regenerate bad tape if possible. If no corrective action is possible pull job.
STOP 1402 ₈	Label B6 "Bermuda Shred" save for next job.

TABLE 2-1. TAPE FORMAT - LOGICAL RECORD

SHRED OUTPUT - HIGH SPEED RADAR FOR GODDARD

B S				Bit 35
Word 1	TIME	OF ARR	IVAL -	u sec.
2			, SUBC	
3	TIME	FOR T	RAP ~μ	sec.
4 5		FIRST 3	6 BITS	
6			EMETR	′
8				
9 10		RAN	IGE ₁	
11 12		AZIM	UTH	
13	*1	ELEV	ATION	
14 15		* 1	0 1	See note
16 17 18 19			D 36 BIT EMETR	
20				
21 22	ļ	RAN	IGE ₂	
23 24		AZI	иитн	
25 26	*2	ELEV	ATION	
27		* 1	0 1	See note

ii S	•							Bit 35
	Т	IME	OF	ARR	RIVA	۸L ~	μs	ec.
Ī	DE 0 (NT, 2 4 0				٧.
		TIM	E F	OR T	RA	P ~μ	ιse	с.
	0 0 0 0		0 0 0	0 0 0 0	0 0 0	0	0 0 0	0
		J		RAN	3E3			
				AZIN	UT	Н		
١		*3	E	LEV	ATI	ON		
١			*	1	1	0	See	note
	0 0 0	0 0 0	0 0 0	0 0 0			0 0 0	0 0
	0	0	0	0				
				RAN	GE	4		
			/	AZIM	IJΤI	14		
	_	*4	E	LEV	ΑТ	ON		
			*	1	1	0	See	note

NOTES: 01 = Cape, 02 = Grand Bahamas, 03 = San Salvadore.

 $\star = 1000$ mile recycle bit for range.

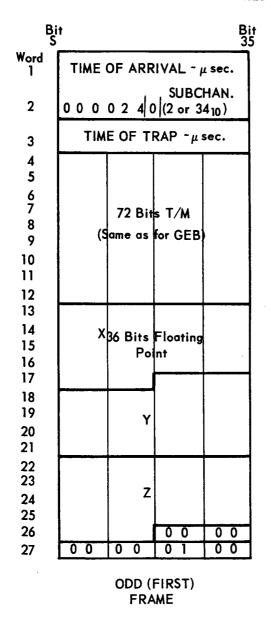
* = On Track bit.

1 = On Track.

0 = Off Track.

TABLE 2-2. TAPE FORMAT - LOGICAL RECORD

SHRED OUTPUT - IP7094



Bi	t .			Bi 3.
	TIME	OF ARR	IVAL ~	μ sec.
	000	0 2 4	SUBC 0 (2 or	HAN. 34)
	TIME	OF TR	AP ~μs	ec.
		×		
		Ý		
		ż		
	X i X X · X	X X X X	S X X X X 0	X X X X 0 0
	0 0	- TI	ME	0 0
		CHECI	k sum	
	0 0	0 0	0 0	0 0

EVEN (SECOND)
FRAME

TABLE 2-3. TAPE FORMAT - LOGICAL RECORD SHRED OUTPUT - GE BURROUGHS

TIME OF ARRIVAL - μsec. DECREMENT, , SUBCHAN. 0 0 0 0 2 4 0 (1 or 33 10) TIME FOR TRAP - μ sec. TIME RETRO FIRE RETRO FIRE At 42 43 44 45 46 47 48 RETRO FIRE DISCRETE AX X X X RETRO X X X X RETRO X X X X X RETRO FIRE X X X X X X X X X X X X X		Bit S				it 5	B S		
2	Word 1	TIME	OF AR	RIVAL	-μsec.			TIME	OF
3	2								
5 6 7 8 8 9 41 42 43 44 45 46 47 48 10 49 11 57	3	TIME	FOR T	RAP ~	u sec.			TIME	E FC
6 7 8 9 41 42 43 44 45 46 47 48 10 11 57 12 65 66 67 68 69 70 71 72 13 14 X X X X 15 16 17 X X X X 18 19 20 X X X X 21 22 23 X X X X X 24 25			ECT			i :		S X	Х
9	6 7			RE Fl	TRO RE	i		S X	X
11 57	9		43 44	45 46	47 48				
13 14 X X X 15 (X) 16 17 X X X X 18 19 20 X X X X 21 22 23 X X X X 24 X								хх	X
14 15 16 17 X X X 18 19 20 X X X 21 22 23 X X X 24 25		65 66			71 72			XX	Х
16 17 18 19 20 X X X 21 22 23 X X X 24 25	14	X	XX		_				CI
19 20 X X X 21 22 23 24 25 X X X X	16	<u> </u>			_				
20			Y						
22 23 24 25 X X X X	20 21	Х			_				BI-
25 X	23	X			_				
26 00000000	25		х						
		L			1	1			
27 0 0 0 0 1 0 0	27	0 0	0 0	0 1	0 0]		<u> </u>	

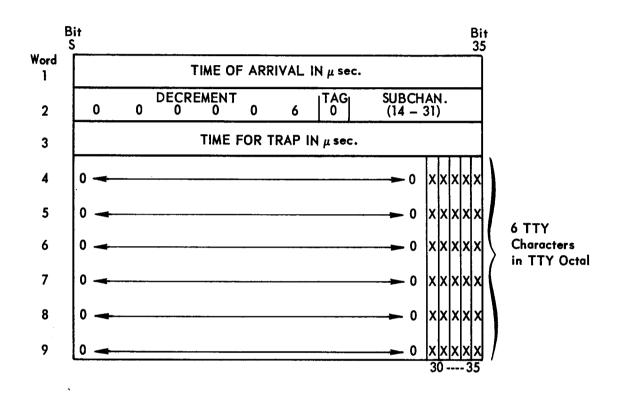
11			35		
TIME OF ARRIVAL ~μsec.					
	MENT, 0 2 40				
TIME	FOR T	RAP ~μ	sec.		
S X	ΧX Ÿ				
S X	X X Ż				
TIMI X X X X	= - X X X X X	X X X X X X	X X X X X X		
	CHEC	k sum			
	BIT PA	TTERN			
	0	10	0 0		

Bit

ODD (FIRST) FRAME

EVEN (SECOND) FRAME

TABLE 2-4. TAPE FORMAT - LOGICAL RECORD SHRED OUTPUT - LOW SPEED TTY



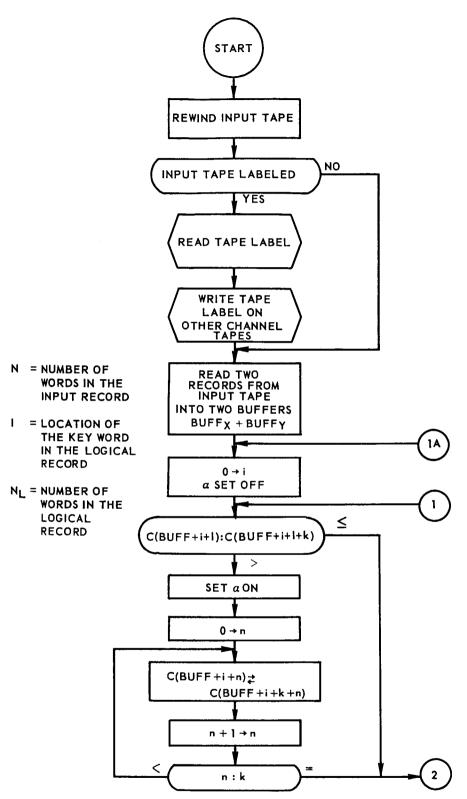


FIGURE 2-9. SORT PROGRAM FLOW CHART (Sheet 1 of 5)

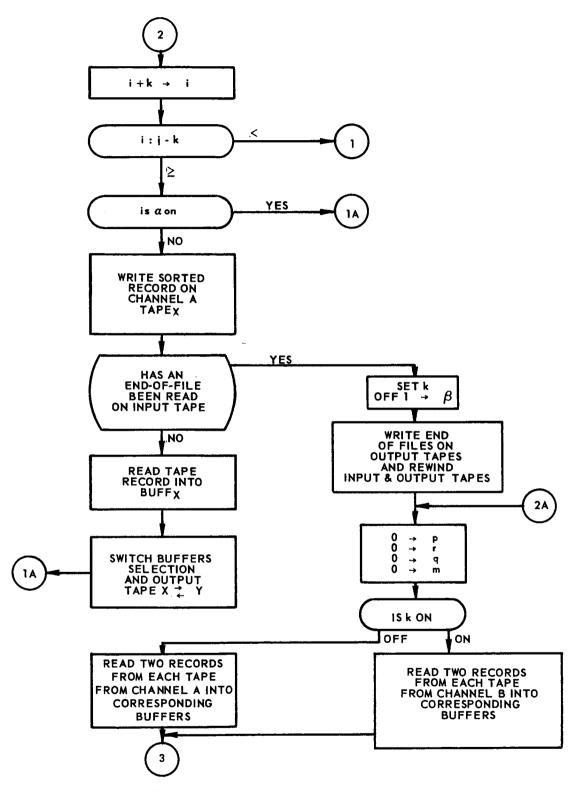


FIGURE 2-9. SORT PROGRAM FLOW CHART (Sheet 2 of 5)

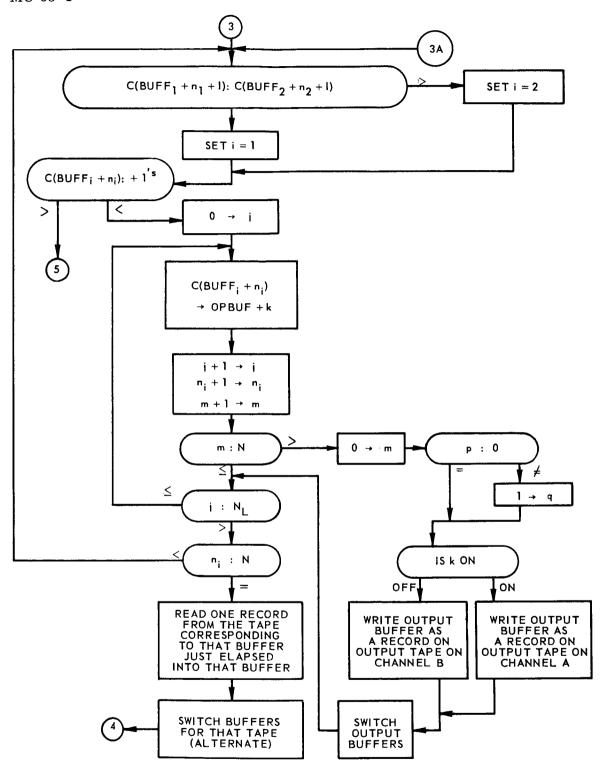


FIGURE 2-9. SORT PROGRAM FLOW CHART (Sheet 3 of 5).

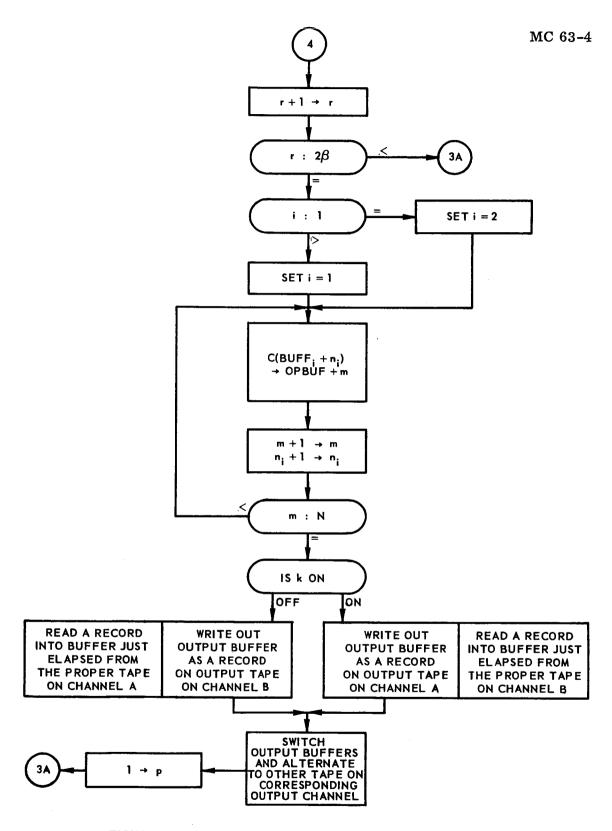


FIGURE 2-9. SORT PROGRAM FLOW CHART (Sheet 4 of 5)

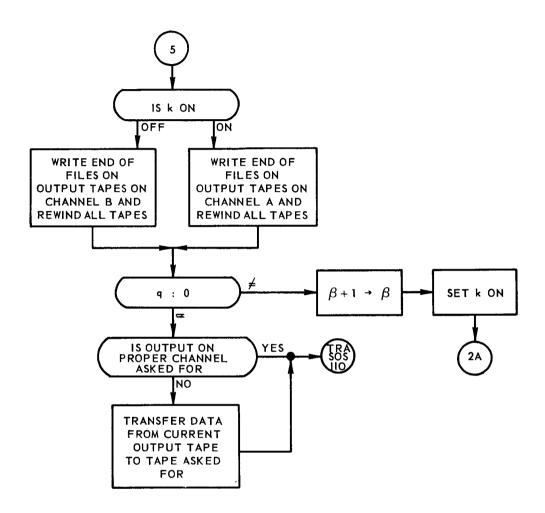


FIGURE 2-9. SORT PROGRAM FLOW CHART (Sheet 5 of 5)

2.11 MERGE PROGRAM

Merge combines several sorted binary input tapes containing radar information to obtain a single input tape—with the information arranged in sequence according to the keyword. Its main purpose is to merge a tape containing high-speed data used by the launch programs with a tape containing low-speed data for orbit and reentry portions of the flight profile, thus obtaining a single tape containing simulated data of an actual Mercury launch-orbit-reentry mission.

The flow chart for Merge is shown in Figure 2-10.

2.11.1 Input Requirements

Merge is written in a general fashion and combines up to nine input tapes. One of the input parameters required by the program is the number of input tapes to be merged. The number is entered using an equals card to equate a parameter to the number of tapes. The program also has the capability to merge tapes of different record sizes (i.e., Merge handles variable record sizes between different tapes). The same is true for variable size logical records and for the location of the key word within the logical record between the tapes. The key word in the logical record of one tape can be the first word, in the next tape the key word can be the last word, etc. This does not mean that the record size, the location of the key word, or the size of the logical record is variable within one given tape. This means only that the size of the logical record and the location of the key word is variable between tapes. Therefore, input to the Merge program will be equals cards which indicate to the program the size of the tape record for that particular tape, its logical record, and the location of the key word. Each tape requires these three input cards. If nine input tapes are used, 27 parameters are needed to describe the essential quantities needed by the program.

Another parameter indicates when the input tapes are labeled. The program does not have the capability to merge tapes having labels with tapes not having labels; i.e., unlabeled tapes and labeled tapes cannot be merged.

2.11.2 Output Requirements

The Merge program can present any desired record size in its final output tape. The record size is fixed by an equals card.

2.11.3 Method

The Merge program first looks at the parameter which indicates the number of input tapes; then looks at the input parameter to determine if the input tapes are labeled. If they are labeled, the program reads the label from each

input tape and writes one of these labels on the output tape. Two input records are then read from each tape—the assumption is that the input tapes are now all sorted within themselves.

A comparison is made between words of all the logical records to determine which one is the low-order word. This key word, together with its logical record, is transferred to the output buffer. The Merge program continues to read records of each tape, comparing the key words of each tape and storing the logical records in sequence in the output buffer. The contents of the output buffer is written on the output tape when the record size is reached. This record is written on the proper tape as indicated by the logical setting. The program continues this process until it has read an end-of-file indication on all input tapes. The program now fills the last unused record buffer with high padding (a word containing all 1's with a plus sign), the reason is that in a sort operation this would be the highest magnitude. An end-of-file is then written at the end of the output tape, the output tape is rewound, and a transfer is made to SOS for completion.

2.11.4 Usage

- Tape Set-Up: A1 SOS Sys. tape a) List Tape **A2** Job Tape **A**3 A6 Blank For Output B1 Blank **B**2 Blank Sorted High Speed B3**B4** Sorted Low Speed
- b) Sense Switch Setting—none.
- c) Key Setting—none.
- d) Operating Instructions—clear machine and load tape.
- e) Halts:

HPR 33333 ₈	Tape read error, press START to continue; if error persists regenerate bad tape if possible. If no corrective action is possible pull job
STOP 1402 ₈	Write end-of-file on A2. Rewind A6 and IBTD first few records of A6 onto A2. Write end-of-file A2 and list all files.

Label and save A6 per instruction sheet

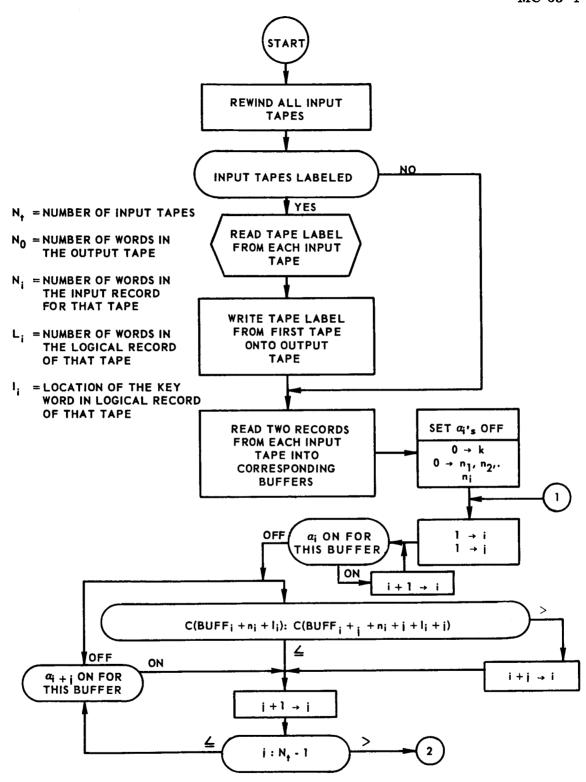


FIGURE 2-10. MERGE PROGRAM FLOW CHART (Sheet 1 of 2)

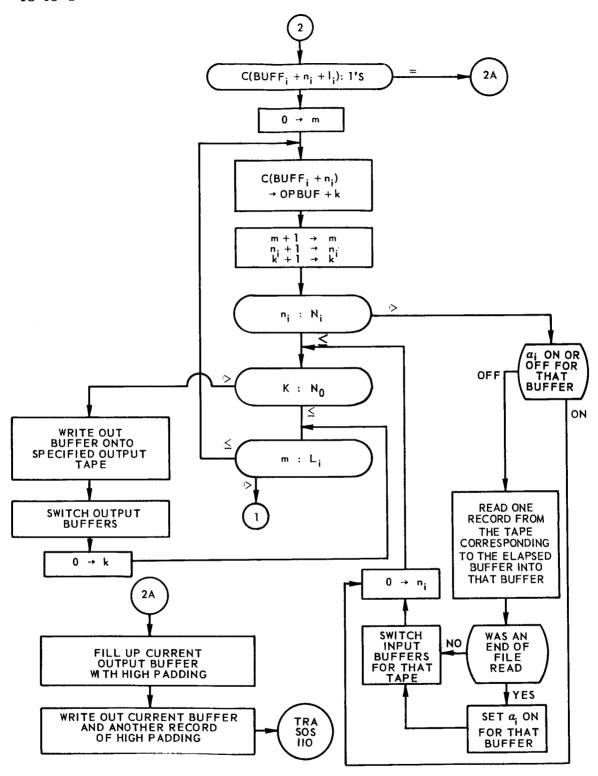


FIGURE 2-10. MERGE PROGRAM FLOW CHART (Sheet 2 of 2)

2.12 SIMULATED INPUT/OUTPUT CONTROL PROGRAM (SIC)

SIC accepts the output of the Shred program (a data preparation program) and enters this information into the real-time processing system by maintaining control over monitor program execution and by simulating the Data Communications Channel (DCC) operation.

The SIC program:

- a) Maintains an estimate of simulated elapsed time.
- b) Provides simulated input to the Goddard computers.
- c) Records all output.
- d) Simulates all traps.
- e) Permits the use of Share Operating System (SOS) macros.
- f) Measures machine loading time.

SIC performs these functions in such a way that time constraints on the system are nearly the same as in a real-time situation.

The flow chart for SIC is shown in Figure 2-11; Table 2-5 lists the routines and subroutines used by SIC.

2.12.1 Input Requirements

The formats of the logical records in a tape record have the same basic pattern and are divided by time of arrival. The first three words are control information; the rest of the logical record contains the exact configuration of bits that should be in an input region. For example, the body for teletype contains six words, each containing one teletype character.

The control words are:

Word 1—the time at which the first bit of the first word should arrive in the computer. This time is expressed as a binary integer in milliseconds.

Word 2 Decrement-number of words in body.

Word 2 Address-DCC subchannel to be used.

Word 3—time at which the last bit of the last word should arrive in the computer (i.e., the time of the trap).

Words 4 and continuing to next message—body of the message.

2.12.2 Output Requirements

The format for the output record is similar to the format used for input, except for control words. The first control word of output corresponds to the second word in the input; the second word corresponds to the third input word. The third output word contains the Present Sense Lines (PSLF) mask. The body of the output message is contained in consecutive words, starting with the fourth word. It is possible, however, for the records to be slightly out of actual time sequence because of the need for using a discrete estimate (SIC's clock) for continuous real time.

2.12.3 Method

A timing device in the DCC gives SIC control every millisecond via a trap over subchannel 1. Not every trap, however, causes the entire set of SIC routines to be executed. SIC performs its functions based on a predetermined interval which may not correspond to the 1 millisecond trapping interval. For example, SIC programs may be executed every 3 milliseconds although the trap occurs every millisecond. This procedure is followed to provide simulation flexibility and to adjust computer time to processing requirements.

The input interval used by SIC is established by bypassing N-1 of N traps, where N is the input parameter. The time the computer remains enabled between successive traps is therefore N milliseconds. At the beginning of each run, the length of time which SIC assigns as N milliseconds must be defined.

There is no way for the Mercury program to stop the input without losing it. Input enters the computer based only on its time-of-arrival tag and SIC's simulation of real time. However, the Mercury program can ignore input by disabling the DCC (all DCC instructions must be simulated by SIC) thereby shutting off the trap for the particular subchannel. Instead of giving the command directly, a reference table is used. If operating in a real-time situation, these locations contain normal instruction; when simulating, an STR (store location and trap) instruction which gives control to SIC, is used causing the proper command to be simulated.

For debugging purposes it is desirable to suspend the normal flow of the SIC/Mercury programs and perform debugging operations, such as selective core dumps. The SOS system has many debugging macros useful for this purpose. SIC makes possible the use of these SOS features by packaging all SOS

macros with a pair of off-clock and on-clock subroutines. The off-clock subroutine disables the DCC (with its one-millisecond pulse) and gives control to the programmed debugging macros, or initiates any other action which is to take place during the suspension of time. To restart time, the program calls for a SIC start-clock subroutine which restores conditions to what they were at the time of the off-clock action, restarts the timer and returns control to the program. Mercury programming then continues from the point where SIC/Monitor program operation was suspended.

There are two stops in SIC for initializing the clock setting. One stop is in the initializing block of SIC and the other is in a subroutine that is entered by a TSX with the DCC disabled. Both stops are for defining the time (N milliseconds). The stop in the initializing block also permits a definition of the time to start looking at data, allowing for a step forward into a tape on input so processing begins with pertinent data. The reason for two (or more) stops is that during launch a faster computer is required than during orbit.

During the orbit phase the incoming flow of data is greatly reduced and the computer essentially idles, waiting for new input. For simulation runs this idle time uses valuable computer time and provides nothing in return. Such idle time is employed by a combination of three techniques: (1) SIC is set to bypass N-1 out of N-millisecond traps (where $N \le 35$; N=10 is generally used), (2) the amount of time SIC assigns to the N-millisecond period between traps can be increased, and (3) a special routine called STPTME (step time) is used to move the SIC estimate of time forward to the next trap time if the computer is idling.

All output from DCC generated by the Monitor programs are recorded by SIC. This recording is accomplished by setting up a signal which SIC recognizes as a trap when Monitor starts to write output. When SIC recognizes the trap, it removes the data from the output region, classifies and time tags it, stores it in an output region and sets up the next trap. The process is continued until a signal is received from Monitor indicating that no more output is available for the particular subchannel. When filled, the output regions are recorded on tape; they are then reused.

2.12.4 Usage

a) Using SIC to Simulate DCC Instructions

Certain instructions cannot be performed normally without using the DCC; these instructions must be simulated. This is done by using XEC 0 instructions in the Mercury Monitor, where 0 contains STR when using SIC, and

the actual instruction when SIC is not being used. A table of such instructions is shown below.

Location	Normal Instruction	SIC Instruction Used to Call Subroutine to Simulated Normal Instruction
0	RCT	STR 0, 0, 1
0	ENB 0, T	STR 0, T, 2
0	PSLF 0, T	STR 0, T, 3

Both the SIC instruction and the STR can contain useful tags.

Effects of the various instructions are:

- 1) RCT: (XEC 0, T₁) (0, T₁: STR 0, 0, 1):
 - (a) Enables all channels according to the last enabling mask.
 - (b) If given after a data trap, and if more traps are waiting, the new trap is simulated as coming from the location of the XEC + 1 and all channels are inhibited.
- 2) ENB: (XEC 0, T_1) (0, $T : STR 0, T_2$, 2):
 - (a) Enables all channels according to C (0, T₂, 2); C(0, T₂) becomes the new enabling mask.
 - (b) The situation stated in 1(b) above also applies here.
- 3) PSLF: (XEC 0, T₁) (0, T₁: STR 0, T₂, 3):
 - (a) Subchannels of the DCC (simulated) are enabled according to C (0, T_2) the next time they are checked.
 - (b) A PSLF does not enable the channel. Traps which occur on a subchannel before an enabling of the subchannel by a PSLF are not remembered.
- b) Use of SIC Debugging

SIC permits the use of standard SOS debugging macros but they must be sandwiched by SIC subroutines, stop-clock and start-clock, which suspend and restart the special clock. The net effect is that time is suspended during the executions of SOS debugging macros and restarted upon the completion of the executions.

c) Calling Sequence:

The calling sequence for the SIC program is:

STL TRA	OFCLK	(MACRO)
IIM	SOS DEBUG MACROS	
\mathtt{STL}	ONCLK	(MACRO)
TRΛ		•

TABLE 2-5. INDEX OF ROUTINES AND SUBROUTINES USED IN SIC

	NAME	CALLING SEQUENCE	PURPOSE AND REMARKS
1.	SGSTRT	GO Card	Initialize for SIC run
2.	XAA	XEC STR Y,O,Z	Simulate action of DCC when given an RCT, ENB or PSLF command
	XAB XAD XAG	XEC STR Y,0,Z = 1 XEC STR Y,0,Z = 2 XEC STR Y,0,Z = 2	RCT ENB With mask in location y PSLF With mask in location y
3.	CTAA CTAA2	SIC clock trap — No Bypass SIC clock trap — Bypass N-1 of N	Simulator Input/Output Control (SIC)
4.	GSV	SXD GSVS, 4 TSX GSV, 4	Save conditions
5.	GRTN	TSX GRTN, 4	Restore conditions
6.	СТС	C(XB)=Subchannel Identi- fication TSX CTC, 4	Record data leaving computer via DCC
7.	GRD	TSX GRD, 4	Read record of SIC input
8.	XAH or OFCLK	STL XECS + 11 TRA XAH	Disable SIC clock and setup for SOS Debug Macro
9.	XAK or ONCLK	STL XECS + 11 TRA XAK	Restore computer for Mercury programs and turn on SIC's clock
10.	TARA	TSA TARA, 4	Reverse the PSLF mask used by Monitor for use with SIC — (SIC looks at it backwards because of an early misconception)
11.	STPTME	TRA STPTME	Skip time ahead if possible
12.	SGENDX	TRA* STRBLE or TRA SGENDX	Finish output records and dump core — (used to either wrap up a run)

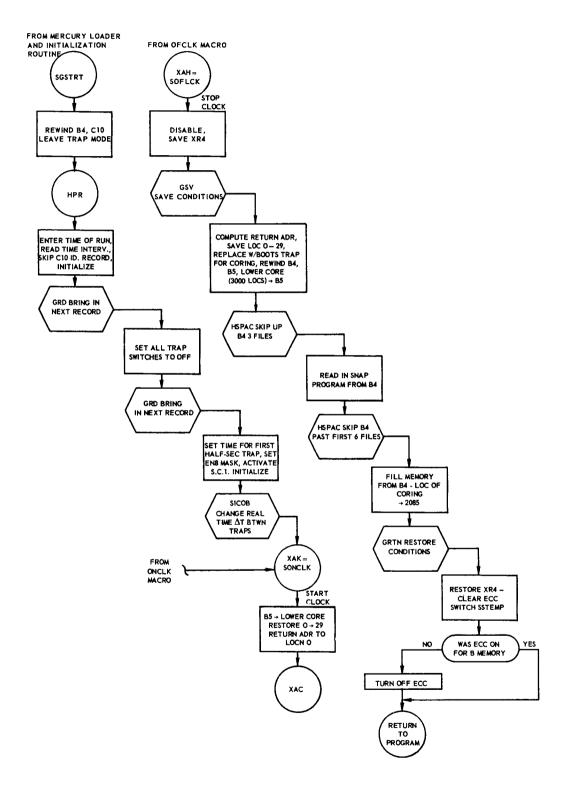


FIGURE 2-11. SIMULATED INPUT/OUTPUT CONTROL PROGRAM (SIC) (Sheet 1 of 5)

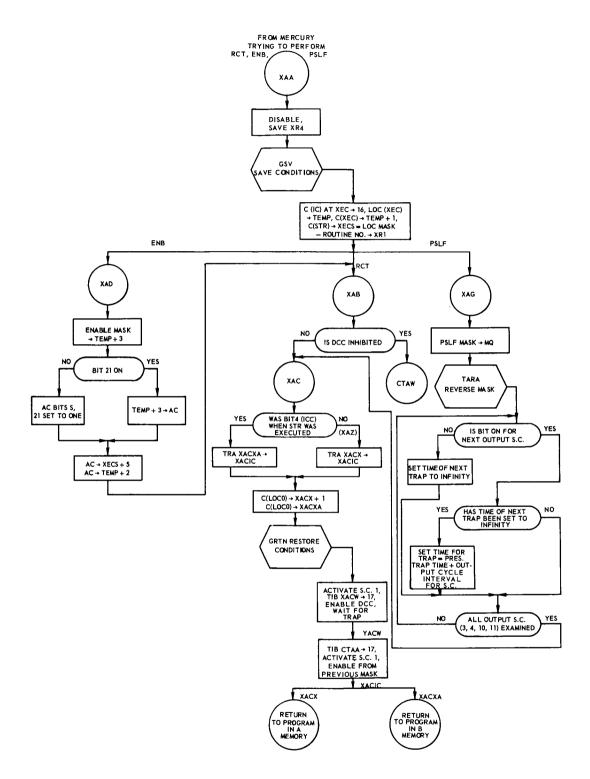


FIGURE 2-11. SIMULATED INPUT/OUTPUT CONTROL PROGRAM (SIC) (Sheet 2 of 5)

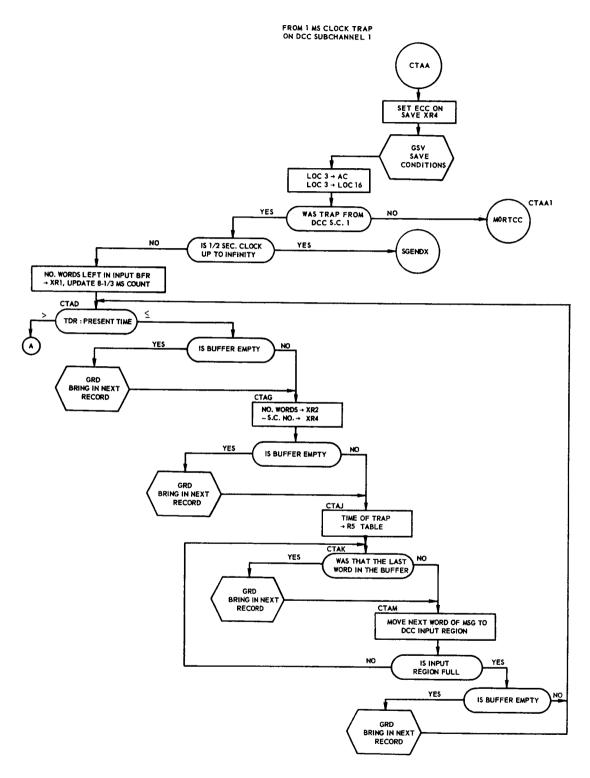


FIGURE 2-11. SIMULATED INPUT/OUTPUT CONTROL PROGRAM (SIC) (Sheet 3 of 5)

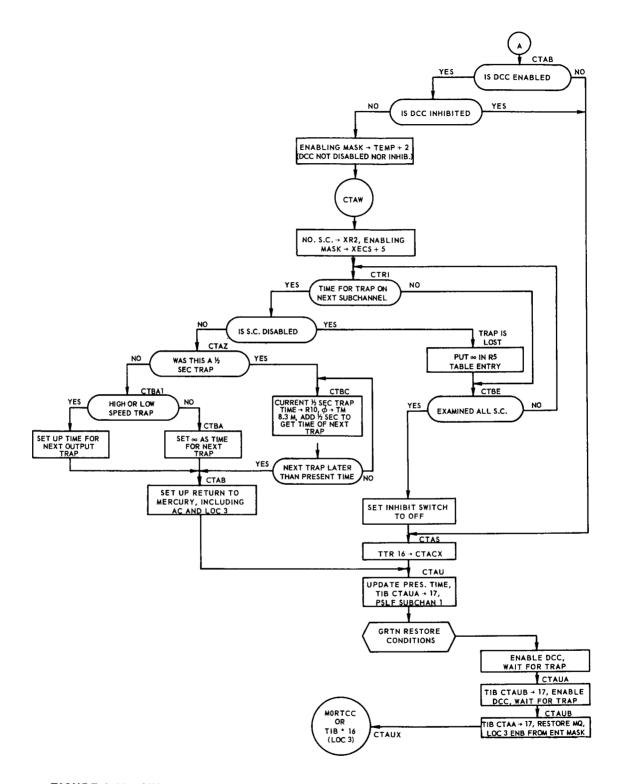
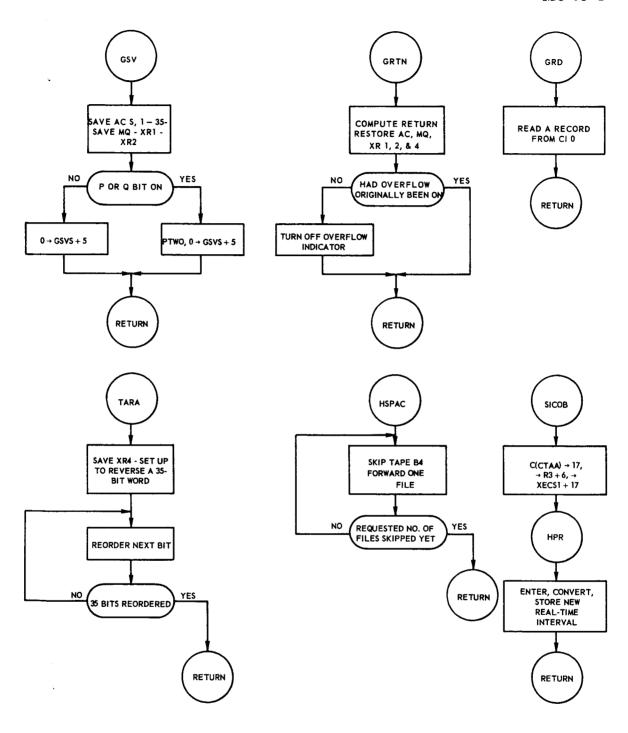


FIGURE 2-11. SIMULATED INPUT/OUTPUT CONTROL PROGRAM (SIC) (Sheet 4 of 5)



SUBROUTINES

FIGURE 2-11. SIMULATED INPUT/OUTPUT CONTROL PROGRAM (SIC) (Sheet 5 of 5)

2.13 CONVERSION OF THE REAL-TIME MERCURY SYSTEM FOR OPERA-TION WITH SIC

The following changes to the real-time Mercury system permit operation under SIC control.

a) Programs Required:

- 1) SIC—simulated input control program
- 2) SGSTRT—this is the first program executed in a SIC run and thus provides the required initialization for SIC
- 3) M0SENT—this program is the equivalent of M0INIT in the realtime system and should contain all the suppressions and indicator settings found in M0INIT. Because time is not initialized externally to the system, it must be preset. The cells AS2 and AS3 should be set equal to the starting simulation time; the total number of minutes in AS2 and the equivalent number of 1/2 seconds in AS3.

b) Changes Required:

- 1) The symbol M0NORG, used by the system loader to transfer control to the system, must be equated to SGSTRT
- 2) The DCC subchannel mask MCACTV must be set to activate those input channels normally activated at the beginning of the phase in which the simulation is to start
- 3) If the system is to be run in the launch phase, the following change must be affected. The high-speed input data blocks for subchannel #1, TMHSGB and TMXSGB, must be moved up in memory as follows:

	ALTER	TMHSGB,	TMXSGB
TMHSGB	BSS BSS BSS	N 24,0 M) can be r	eplaced by BSS M + N
TMXSGB	BSS BSS BSS	N } can be 1 24,0 M	

Where N + M = 8, which represents the words remaining of the original 32-word storage assignment not normally used by the channel. N should be set equal to the number of traps SIC counts

before updating the simulated traps. The maximum count of traps between updatings therefore is eight.

The symbols C1 and C33 located in SIC must be equated to the new decimal location of the tables TMHSGB and TMXSGB in that order.

- 4) If the records on the SIC input tape should change from 198, the cell N (defined within SIC) would then have to be equated to the new record length.
- 5) The decrement of the keys located on the operation console of the 7094 should be considered as a mixed number, that is, an integer and a two-place fraction. Therefore, a decrement setting of 100 (octal) really is 1.00 octal and is the current setting to be used for simulation.

2.14 OPEN LOOP SIMULATION PROGRAM (OLS1) MERCURY CONTROL CENTER

The purpose of the OLS1 program is to furnish data for the real-time operation of displays at the Mercury Control Center. The flow chart for the OLS1 program is shown in Figure 2-12.

2.14.1 Input Requirements

The SOS system is used with the OLS1 program. Input to OLS1 consists of:

- a) B-GE launch data furnished by STL on column binary cards, to be read from tape.
- b) Monitor log tape generated by using STL launch data or actual launch data as input. The log tape blocks physical records of ten 17-word logical records each; each logical record consists of five words of identification information and 12 words of data.

2.14.2 Output Requirements

High-speed output derived from B-GE data drives the displays located at the Mercury Control Center.

2.14.3 Method

The OLS1 program has two distinct parts: one part generates B-GE data displays directly from the STL card; the second part generates Goddard-to-Mercury Control Center data from the log tape. When data from either of these two parts is generated, the OLS1 program reads out that information, on the high-speed lines, to be recorded on the A-Simulator at the Mercury Control Center or to drive MCC displays. The program methods employed during each phase are the same. The OLS1 program merely reads the input data and waits for the 7094 clock to become equal to the time tag on the data. When the times are equal, the data is transmitted from the computer to MCC.

2.14.4 Usage: Operator's Procedures

- a) Call the MCC to arrange to record data on the A-Simulator tape or to drive displays via the high-speed lines
- b) Load the OLS1 program using SOS

- c) Mount STL tape on A6 if B-GE data is desired.
- d) Mount log tape on A7 if normal high-speed output is required.
- e) Mount blank tapes on B3 and B4 for logging high-speed output that is transmitted.
- f) Arrange with the MCC as to which type of data is to be sent.
- g) Enter proper code in keys for program selection. (Key 35 for B-GE data; Key 34 for display data).
- h) Upon receipt of signal from the MCC, press START to begin program.
- i) Save tapes A6, A7, B3 and B4 and label unless otherwise instructed.

All steps are accompanied by an error message printout.

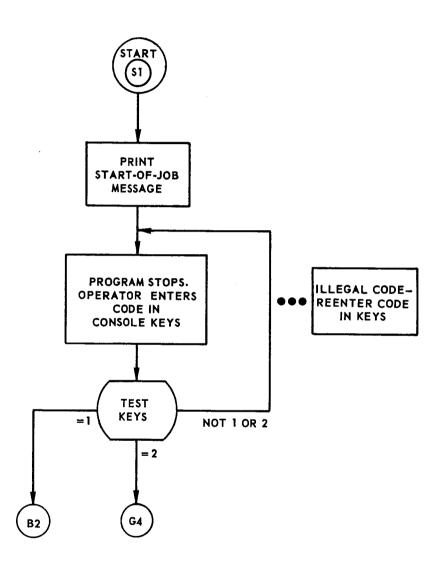


FIGURE 2-12. OLS1 PROGRAM FLOW CHART (Sheet 1 of 4)

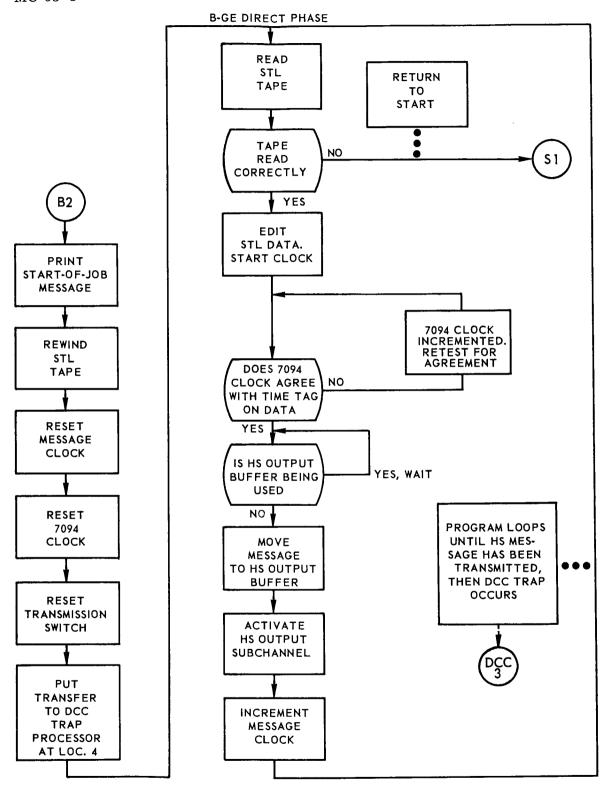
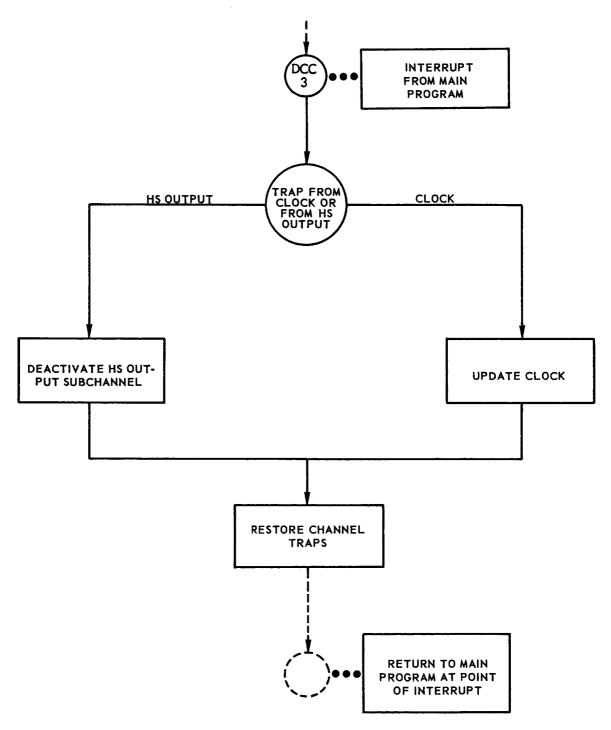


FIGURE 2-12. OLS1 PROGRAM FLOW CHART (Sheet 2 of 4)



TRAP PROCESSOR: DCC TRAPS MAY OCCUR AT ANY POINT IN MAIN PROGRAM

FIGURE 2-12. OLS1 PROGRAM FLOW CHART (Sheet 3 of 4)

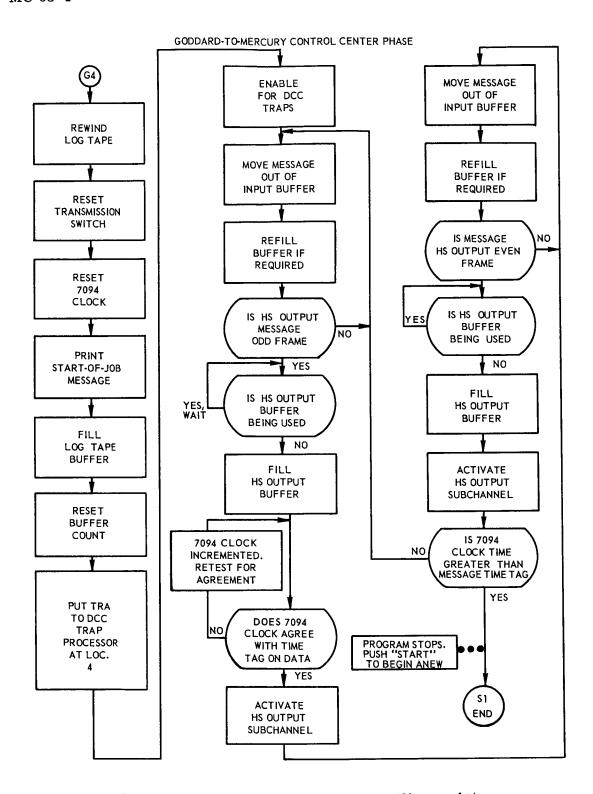


FIGURE 2-12. OLS1 PROGRAM FLOW CHART (Sheet 4 of 4)

2.15 CLOSED LOOP SIMULATION PROGRAM (CLS3) - MERCURY CONTROL CENTER

The purpose of the CLS3 program is to generate a special purpose magnetic tape which contains B-GE to Goddard data, B-GE display data, IP 7094 data, IP 7094 false computer words, and flight flags. This tape will be played on the B Simulator at the Mercury Control Center to supply real-time data for the Goddard computers and the B-GE direct displays during a simulated launch.

The flow chart for CLS3 is shown in Figure 2-13.

2.15.1 Input Requirements

The SOS system is used with the CLS3 program. Input required by the Closed Loop Simulation program are:

- a) STL-furnished B-GE Direct and B-GE-to-Goddard data on punched cards which are to be read from tape.
- b) IP 7094 data produced by the Shred program and read in from tape.
- c) Flight flags on cards furnished by NASA and read from tape. Columns 1-6 of the card contain time in milliseconds since liftoff and columns 8 and 9 contain the flight flag number in decimal.

2.15.2 Output Requirements

The output of CLS3 is a single-record 7-track tape which contains IP 7094 launch data, B-GE display data, B-GE data, a timing track, a flight flag track, and an IP 7094 false computer word track. IP 7094 and B-GE-to-Goddard formats are shown in Figures 2-14 and 2-15, respectively.

2.15.3 Method

The CLS3 program runs in two phases. The first phase under sense switch option extends the IP 7094 Shred tape with false computer words. The second phase reads the three sources of data into core storage and arranges the messages in the block according to the time when they should be read out. Four-block buffers are continuously being filled and written out. The buffers are treated cyclically by the program; the filling and the writing out on tape are always two blocks out of phase with one another.

2.15.4 Usage: Operator's Procedures

- a) Mount STL tape on B3.
- b) Mount IP 7094 tape on A6 if the tape is not to be extended. Mount IP 7094 tape on B6 and a blank on A6 if the tape is to be extended.
- c) Mount flight flag tape on A7.
- d) Mount a blank tape for output on C1 and set the density to 200 bits per inch.
- e) Load program using SOS.
- f) To extend IP 7094 Shred tape, depress sense switch 1 and set the keys to the time at which the extension is to begin. Do not depress sense switch unless the Shred tape is to be extended.
- g) At the completion of the run, remove C1 and send it to MCC.

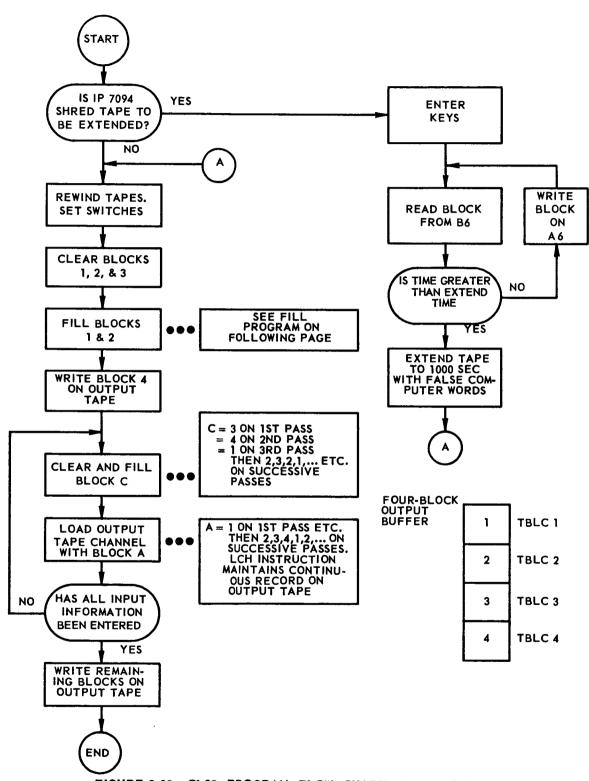


FIGURE 2-13. CLS3 PROGRAM FLOW CHART (Sheet 1 of 2)

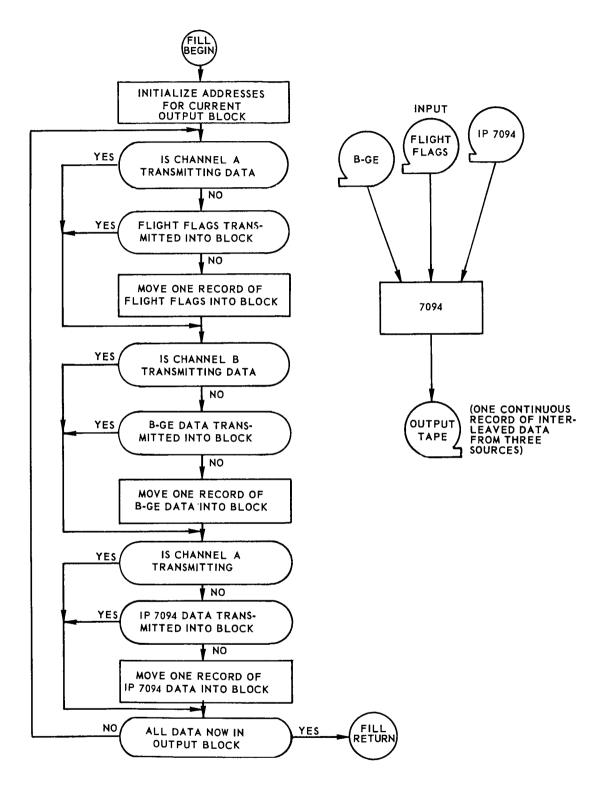


FIGURE 2-13. CLS3 PROGRAM FLOW CHART (Sheet 2 of 2)

EIGHT-BIT WORD TRANSFERS FROM DATA RECEIVER FIRST SUBFRAME

1	1		1	ELE	MET	RY		8
2	9		1	TELE	MET	RY		16
3	17		1	ELE	MET	RY		24
4	25		1	[ELE	MET	RY		32
5	33		1	TEL E	MET	RY		40
6	41		1	ELE	MET	RY		48
7	49		ו	ELE	MET	RY		56
8	57		1	ELE	MET	RY		64
9	65			ELE	MET	RY		72
10	1				A			8
11	9				A			16
12	17			,	A			24
13	25				A			32
14	33		A	36	1	E	3	4
15	5			[3			12
16	13			E	3			20
17	21			E	3			28
18	29				3			36
19	1			(2			8
20	9			(Ĉ.			16
21	17			(2			24
22	25			(32
23	33		С	36	0	0	0	0
24	0	0	0	1	ı	D*	-	5

EIGHT-BIT WORD TRANSFERS FROM DATA RECEIVER SECOND SUBFRAME

		CEIVER SECOND SOBI RA	UVI L.
1	1_1_	D	8
2	9	D	16
3	17	D	24
4	25	D	32
5	33	D 36 1 E	4
6	5	E	12
7	13	E	20
8	21	E	28
9	29	E	36
10	1	F	8
11	9	F	16
12	17	F	24
13	25	F	32
14	33	F 36 1 N	4
15	5	N	12
16	13	N	20
17	21	N	28
18	29	N	36
19	1	CHECKSUM Σ	8
20	9	CHECKSUM Σ	16
21	17	CHECKSUM Σ	24
22	25	CHECK SUM ∑	32
23	33	Σ 36 0 0 0	0
24	0	0 0 1 ID*	5
		<u> </u>	

THE ABOVE CONSTITUTES A COMPLETE MESSAGE FRAME AND IS TRANSMITTED EVERY 400 MILLISECONDS. EACH SUBFRAME CONSISTS OF 192 SERIAL BITS PRECEDED BY A SYNC SIGNAL. THE QUANTITIES REPRESENTED BY A, B, C, D, E, F AND N ARE RESTRICTED INFORMATION AND ARE SPECIFIED IN OTHER DOCUMENTS.

FIGURE 2-14. IP 7094 DATA, MERCURY CONTROL CENTER-TO-GODDARD MESSAGE FORMAT (Sheet 1 of 2)

^{*} SEE NOTE 1 FOR MAKEUP OF ID WORD.

SEE NOTE 2 FOR FORMAT OF BITS 1 TO 72 OF FIRST SUBFRAME IN ABSENCE OF TELEMETRY DATA AND NOTE 3 FOR FORMAT IN ABSENCE OF QUANTITIES A, B, C, D, E, F AND N.

NOTES

1. THE FIVE-BIT IDENTITY (ID) WORD IN EACH SUBFRAME CONVEYS THE FOLLOWING INFORMATION:

DATA FROM IP 7094 HIGH-SPEED BUFFER AND RETRANSMITTER

BIT 1: A ZERO SIGNIFIES IP 709 DATA FORMAT A 1 SIGNIFIES RAW RADAR FORMAT

BIT 2: A 1 SIGNIFIES SECOND SUBFRAME

BIT 3: A 1 SIGNIFIES FIRST SUBFRAME

BITS 4 AND 5: INDICATE SOURCE OF RAW RADAR DATA

- 2. IF NO DATA IS RECEIVED FROM THE TELEMETRY EVENT TRANSMITTING BUFFER, THE HIGHSPEED BUFFER AND RETRANSMITTERS ARE ARRANGED TO TRANSMIT ZEROES IN THE BIT POSITIONS OCCUPIED BY TELEMETRY EVENT DATA BITS 1 THROUGH 40 AND 43 THROUGH 72.
 ONES ARE TRANSMITTED IN POSITIONS 41 AND 42, RESULTING IN ERRONEOUS PARITY FOR
 THE TELEMETRY EVENT DATA MESSAGE.
- 3. IN THE ABSENCE OF DATA QUANTITIES A, B, C, D, E, F AND N, THE COMPLETE MESSAGE FRAME IS TRANSMITTED EVERY 400 MILLISECONDS. TELEMETRY DATA CONTINUES TO BE TRANSMITTED. ZEROES WITH 1'S INTERSPERSED IN CERTAIN POSITIONS ARE TRANSMITTED IN PLACE OF THE MISSING DATA QUANTITIES. THE FOLLOWING BITS APPEAR AS 1'S IN THIS EVENT:

SUBFRAME	EIGHT-BIT WORD NO.	QUANTITY	BIT WITHIN QUANTITY
1	12	A	24
i	15	В	12
i	18	В	36
i	21	С	24
2	3	D	24
2	6	E	12
2	9	E	36
2	12	F	24
2	15	N	12
2	18	N	36
2	19	CHECKSUM	1

FIGURE 2-14. IP 7094 DATA, MERCURY CONTROL CENTER-TO-GODDARD MESSAGE FORMAT (Sheet 2 of 2)

EIGHT-BIT WORD TRANSFERS FROM DATA RECEIVER: FIRST SUBFRAME

					• • •		• •	
1	1		TE	LEM	ETF	RY		8
2	9		TE	LEM	ETI	₹Y		16
3	17		TE	LEM	ETI	RY		24
4	25		TE	L EM	ETF	RY		32
5	33		TE	LEM	ETI	RY		40
6	41		TE	LEM	ETF	₹Y		48
7	49		TE	LEM	ETF	₹Y		56
8	57		TE	LEM	ETF	₹Y		64
9	65		TE	LEM	ETF	₹Y		72
10	1		oisc	RET	E W	ORD)	8
11	1			G				8
12	9			G				16
13	17			G				24
14	1			Н				8
15	9			Н				16
16	17			Н				24
17	1			J				8
18	9			J				16
19	17			J				24
20	1			K				8
21	9			K				16
22	17			К				24
23	1	1	1	1	1	1	1	1
24	0	0	0	1	1D	*		5

EIGHT-BIT WORD TRANSFERS FROM DATA RECEIVER: SECOND SUBFRAME

		•						
1	1			L	-			8
2	9			L				16
3	17			L				24
4	1			N	4			8
5	9			٨	4			16
6	17			N	4			24
7	1			N	ı			8
8	9			N	l			16
9	17			١	ı			24
10	1		CH	IECI	(SU)	И		8
11	9	9 CHECKSUM				16		
12	17	17 CHECKSUM				24		
13	1	1	1	1	1	1	1	1
14	1	1	1	1	1	1	1	1
15	1	1	1	1	1	1	1	1
16	1	1	1	1	1	1	1	1
17	1	1	1	1	1	1	1	1
18	1	1	1	1	1	1	1	1
19	1	1	1	1	1	1	1	1
20	1	1	1	1	1	1	1	1
21	1	1	1	1	1	1	1	1
22	1	1	1	1	1	1	1	1
23	1	1	1	1	1	1	1	1
24	0	0	0	1	I	D*		5

THE ABOVE CONSTITUTES A COMPLETE MESSAGE FRAME AND IS TRANSMITTED WITH AN INTERVAL OF 500 \pm 100 MILLISECONDS BETWEEN THE START OF ONE MESSAGE AND THE START OF THE NEXT MESSAGE. EACH SUBFRAME CONSISTS OF 192 SERIAL BITS PRECEDED BY A SYNC SIGNAL. THE QUANTITIES REPRESENTED BY G, H, J, K, L, M AND N ARE RESTRICTED INFORMATION AND ARE SPECIFIED IN OTHER DOCUMENTS.

SEE NOTE 2 FOR FORMAT OF BITS 1 TO 72 OF FIRST SUBFRAME IN ABSENCE OF TELEMETRY DATA AND NOTE 3 FOR FORMAT IN ABSENCE OF QUANTITIES G, H, J, K, L, M AND N.\

FIGURE 2-15. B-GE DATA, MERCURY CONTROL CENTER-TO-GODDARD MESSAGE FORMAT (Sheet 1 of 2)

^{*} SEE NOTE 1 FOR MAKEUP OF ID WORD.

NOTES

1. THE 5-BIT IDENTITY (ID) WORD IN EACH SUBFRAME CONVEYS THE FOLLOWING INFORMATION:

DATE FROM B-GE HIGH-SPEED BUFFER AND RETRANSMITTER.

- BIT 1: ALWAYS A ZERO
- BIT 2: A 1 SIGNIFIES SECOND SUBFRAME
- BIT 3: A 1 SIGNIFIES FIRST SUBFRAME
- BIT 4: ALWAYS A ZERO
- BIT 5: ALWAYS A ZERO
- 2. IF NO DATA IS RECEIVED FROM THE TELEMETRY EVENT TRANSMITTING BUFFER, THE HIGH-SPEED BUFFER AND RETRANSMITTERS ARE ARRANGED TO TRANSMIT ZEROES IN THE BIT PO-SITIONS OCCUPIED BY TELEMETRY EVENT DATA BITS 1 THROUGH 40 AND 43 THROUGH 72. ONES ARE TRANSMITTED IN POSITIONS 41 AND 42, RESULTING IN ERRONEOUS PARITY FOR THE TELEMETRY EVENT DATA MESSAGE.
- 3. IN THE ABSENCE OF DATA QUANTITIES G, H, J, K, L, M AND N, THE COMPLETE MESSAGE FRAME IS TRANSMITTED EVERY 650 MILLISECONDS. TELEMETRY DATA CONTINUES TO BE TRANSMITTED. ZEROES WITH 1'S INTERSPERSED IN CERTAIN POSITIONS ARE TRANSMITTED IN PLACE OF THE MISSING DATA QUANTITIES. THE FOLLOWING BITS APPEAR AS 1'S IN THIS EVENT:

SUBFRAME	EIGHT-BIT WORD NO.	QUANTITY	BIT WITHIN QUANTITY
1	13	G	24
1	16	н	24
1	19	j	24
1	22	K	24
2	3	L	24
2	6	M	24
2	9	N	24
2	10	CHECKSUM	1

FIGURE 2-15. B-GE DATA, MERCURY CONTROL CENTER-TO-GODDARD MESSAGE FORMAT (Sheet 2 of 2)

2.16 CLOSED LOOP SIMULATION PROGRAM (BCLS2) BERMUDA TO GODDARD

BCLS2 writes a continuous-record, 3-channel tape which, when read by the B-Simulator, is converted to Verlort and AN/FPS-16 radar signals to be recorded by the A-Simulator tape drive. When this tape is transported to Bermuda, it can be read by the operational data recorder and the signals generated can be used as radar input to Goddard.

The flow chart for BCLS2 is shown in Figure 2-16.

2.16.1 Input Requirements

Input to BCLS2 is a Bermuda SIC input tape containing Verlort and AN/FPS-16 radar data blocked in 198-word physical records. The logical records have three identification words each and a variable number of data words. Both AN/FPS-16 and Verlort radar input records are 15 words long, including identification. The SOS system must be used with this program.

2.16.2 Output Requirements

The B-Simulator output tape is a single-record, 3-track tape. The three tracks are Verlort and AN/FPS-16 radar messages and the timing track. The radar message format is shown in Figure 2-17.

2.16.3 Method

The BCLS2 program runs in two phases. The first phase reads the SIC tape and duplicates only the radar data. The second phase reads the duplicated SIC tape and extracts the Verlort and AN/FPS-16 radar data from it. The program then writes this data, along with a timing track, on a timed output tape. BCLS3 has a 4-block output buffer which is treated cyclically. The program is always filling one block at the same time another is being written on tape; the filling and the writing on tape are always accomplished two blocks out of phase with one another.

2.16.4 Usage: Operator's Procedures

- a) Mount Bermuda SIC tape on B5.
- b) Mount a Blank on A5.
- c) Mount intended output tape on B3.

- d) Load program using SOS.
- e) Printout indicates when program is finished.
- f) Remove B3, label, and send to the Mercury Control Center.

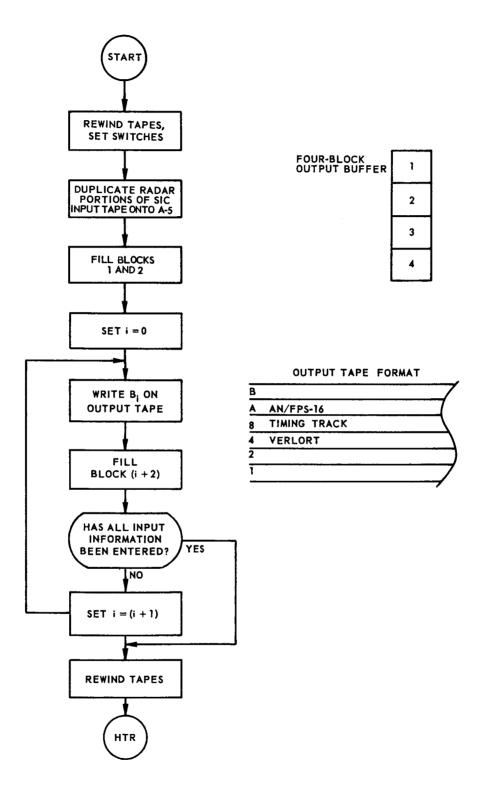


FIGURE 2-16. BCLS2 PROGRAM FLOW CHART

2.17 READ LOW-SPEED SIC TAPE (RLSST)

RLSST generates SIC input tapes. The prime purpose of the program is to convert the TTY message data contained on the SIC tape to BCD. In addition, a BCD tape containing the simulated radar observations is prepared.

2.17.1 Input Requirements

The binary SIC tape constitutes the only input of this program. The input tape consists of simulated Verlort and AN/FPS-16 radar observations grouped into physical records, 198 words in length. Records are composed of twenty-two, 9-word logical records.

2.17.2 Output Requirements

RLSST utilizes SE9OU2 (DNOUT) to prepare the BCD output tape. Access to all information included in the radar messages can be obtained providing it is tape listed.

2.17.3 Method

The program reads and extracts radar data from the SIC tape. Each logical record on the SIC tape contains a subchannel number that identifies the line over which the data is being sent. The data is packed into the storage buffer for any given subchannel. When a 34-word message is completed, the data is converted to BCD and written on the output tape.

The program employs the SOS system, with an additional blank on B-3 (output), and the input on C-10. Netiher the sense switches nor the console keys are required for operation of this program.

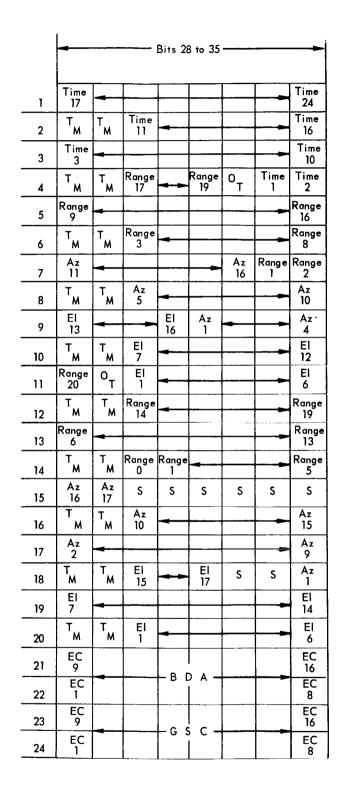


FIGURE 2-17. HIGH SPEED BERMUDA INPUT FORMAT FROM DCC

Section 3

UTILITY PROGRAMS

Utility programs complement monitor and computational programs in the Mercury Tracking System. As such, they are service programs which produce input tapes, process output tapes, and satisfy various specialized needs.

Many utility programs are recorded on a C1 utility tape to provide availability of utility programs as needed. Each such utility program appears on the C1 tape as a single record; loading instructions are at the beginning of each record.

Program selection and loading from the C1 utility tape is accomplished with the use of a call card. The information contained on this type of card positions the tape to the desired record and initiates the loading action. Before positioning and after loading, a call card rewinds the C1 tape. Once loaded, each program is identified on the on-line printer before its execution.

3.1 PROGRAM TO PRINT SELECTED DCC SUBCHANNEL INPUT-OUTPUT DATA FROM A MERCURY LOG TAPE (MXCHER)

MXCHER reads the B6 log tape and selects and prepares for off-line printing in octal the input-output data identified with selected DCC subchannel numbers. An option is provided to permit either searching the tape for several subchannel numbers in one pass or of searching separately for each requested number. In the first method, entries are printed in the order in which they appear on tape; in the second, all entries for a given subchannel are printed together.

The flow chart for MXCHER is shown in Figure 3-1.

3.1.1 Input Requirements

Input to MXCHER includes the B6 log tape produced during a Mercury run.

3.1.2 Output Requirements

Output from MXCHER is produced on A2. Entries are printed vertically across the page, six to a page. Marginal numbering is provided, and on- and off-line comments are produced as required.

3.1.3 Method

The first record on the B6 tape is tested to ensure readability and to set the density mode. When processing, tests are made for redundancy, EOF, EOT, and blank tape conditions. On-line comments are produced as required, and all conditions, except an output redundancy, result in a program stop.

3.1.4 Usage

MXCHER is entered with a call card read on line.

- a) Storage Required—2101 locations
- b) Operating Notes:
 - 1) Sense Switches are not used
 - 2) Entry Keys:

S-when up, subchannels are printed in the order in which they

occur. When down, a separate pass is made for each selected subchannel.

1 to 35—correspond to subchannels to be printed

c) Stops: (all stops are accompanied by on-line print)

20044_{8}	HPR	401 ₈ - Waiting for request word in keys
200318	HPR	402 ₈ - Cannot read B6
200368	HPR	403_8 - B6 is a BCD tape
200548	HPR	404_8 - No entry has been made in keys
202658	HPR	405 ₈ - B6 was running away
203038	HTR	406_8 - FINAL STOP (press start to do another)
203558	HPR	407 ₈ - Redundancy or wrong word count (see on-line print)
203578	HPR	410 ₈ - A2 End of tape

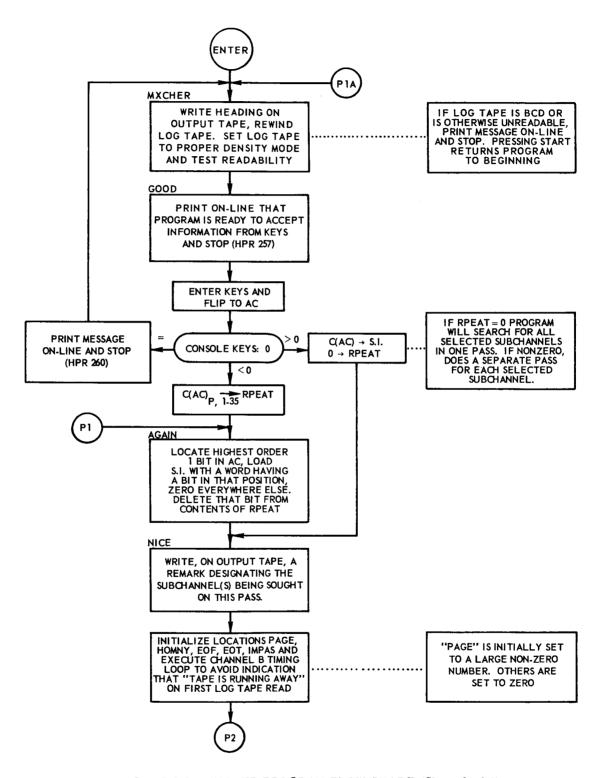


FIGURE 3-1. MXCHER PROGRAM FLOW CHART (Sheet 1 of 6)

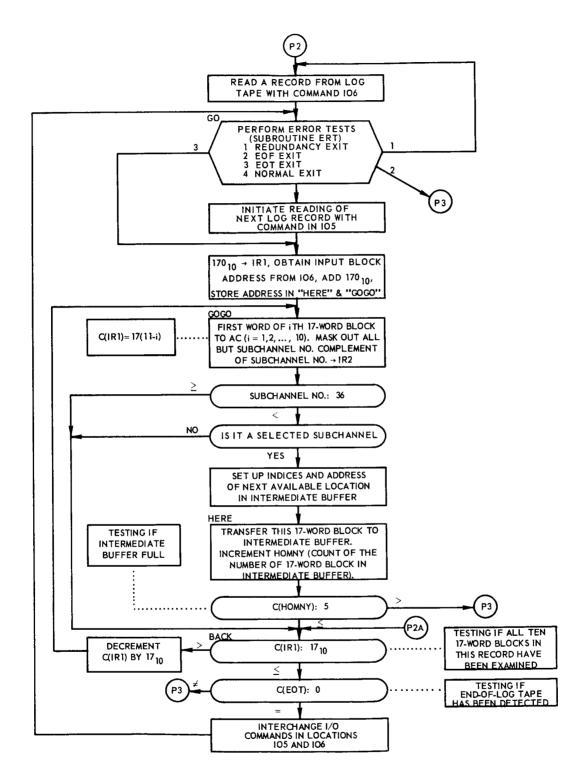


FIGURE 3-1. MXCHER PROGRAM FLOW CHART (Sheet 2 of 6)

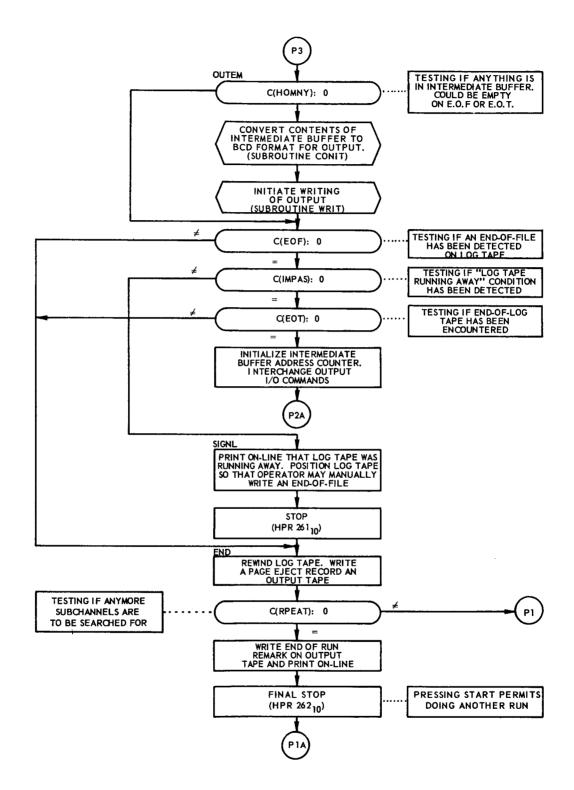


FIGURE 3-1. MXCHER PROGRAM FLOW CHART (Sheet 3 of 6)

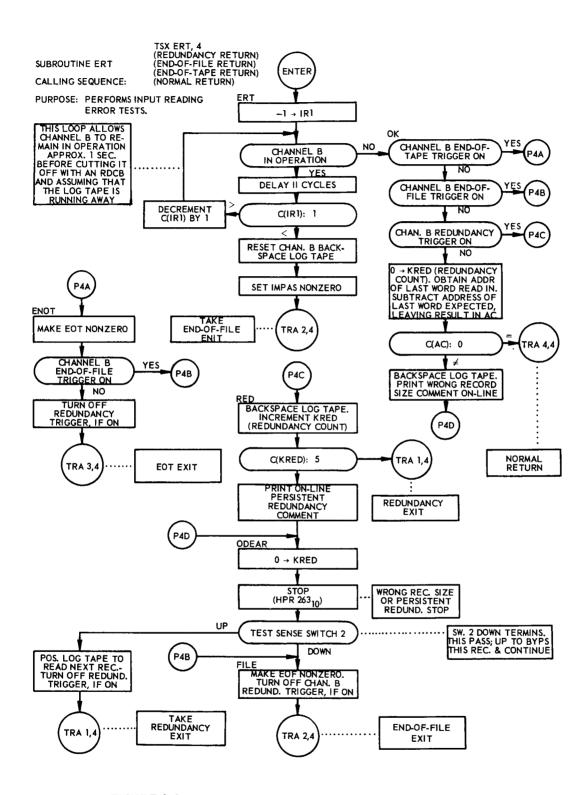


FIGURE 3-1. MXCHER PROGRAM FLOW CHART (Sheet 4 of 6)

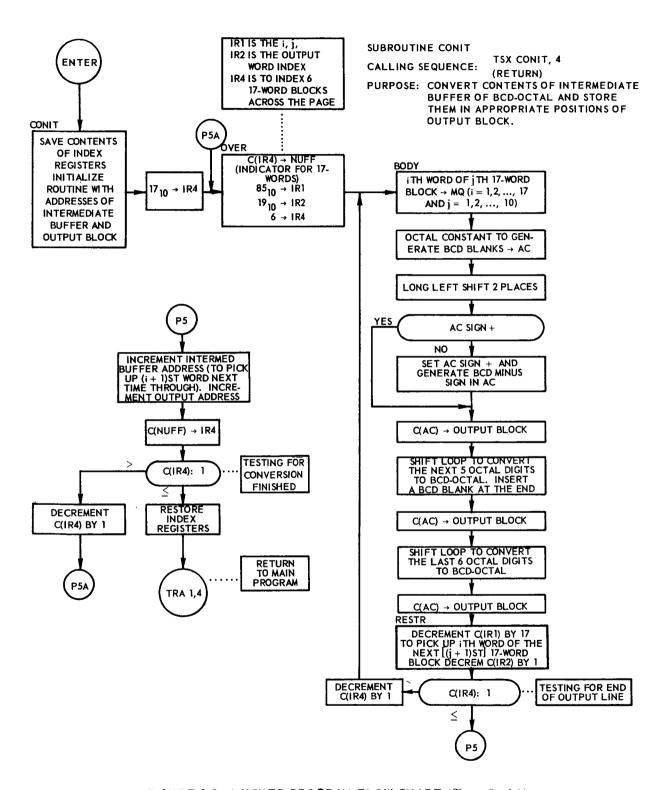


FIGURE 3-1. MXCHER PROGRAM FLOW CHART (Sheet 5 of 6)

SUBROUTINE WRIT

CALLING SEQUENCE: TSX WRIT, 4
(RETURN)

PURPOSE: PERFORM ERROR TESTS ON LAST OUTPUT TRANSMISSION AND INITIATE CURRENT OUTPUT OPERATION.

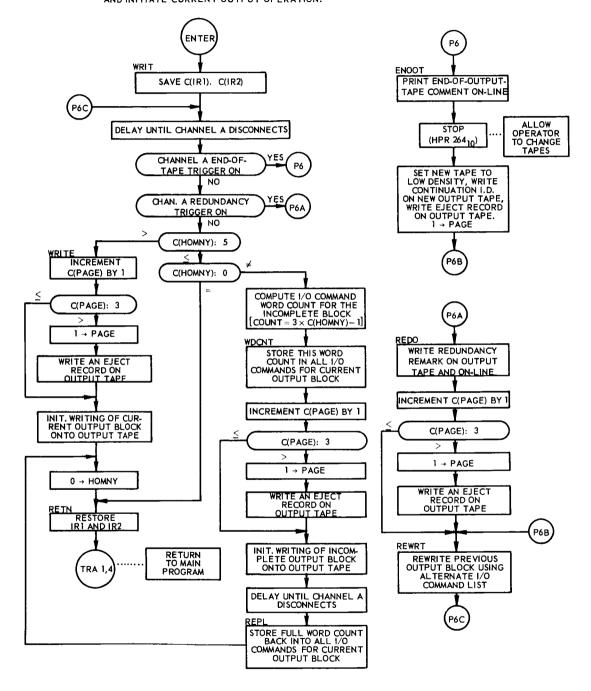


FIGURE 3-1. MXCHER PROGRAM FLOW CHART (Sheet 6 of 6)

3.2 PROGRAM TO PRINT MERCURY LOG TAPE IN OCTAL (MXPOCL)

MXPOCL reads the B6 tape produced from a Mercury run and prepares the recorded information for off-line printing in octal. An option is provided to permit: 1) printing the contents of the entire tape, 2) begin printing just before the liftoff indication, or 3) printing only those entries time tagged within a selected time interval.

The flow chart for MXPOCL is shown in Figure 3-2.

3.2.1 Input Requirements

Input to MXPOCL includes the B6 tape produced during a Mercury run.

3.2.2 Output Requirements

Output from MXPOCL is produced on A6, the tape used for off-line printing.

Output consists of ten 17-word blocks. The blocks are printed vertically, and successive blocks are spaced across the page, left to right, in two rows per log tape record. Marginal numbering of lines is provided to facilitate reading. Each page of output represents two 170-word records from the log tape.

3.2.3 Method

The first record on the B6 tape is tested to ensure readability before processing begins; however, no attempt is made to interpret this information. Following the transfer of blocks from B6 to A6 and from A6 to the printer, redundancy tests are made and EOF and EOT conditions are tested. Appropriate comments are entered where necessary.

3.2.4 Usage

MXPOCL is available in squoze or absolute binary; however, neither the squoze deck nor column binary deck produced from it may be read off line. To run the binary version, the log tape must be placed on B6 and a blank tape on A6. To run the squoze version, SOS must be on A1, blanks on A2, B1, and B2, in addition to the tapes on B6 and A6.

- a) Storage Required—2511 locations
- b) Error Codes—Program stops occur each time an EOF, EOT, or persistent redundancy is detected. On-line messages indicate the cause of the stop and the action to be taken.

- c) Special Usage: The user has the option of operating this program in one of three modes controlled by Sense Switch 2 and the console input switches:
 - 1) To print the entire log tape or any portion of it from the beginning, SENSE Switch 2 must be up; the keys are not examined.
 - 2) To print all information from the log tape, following the first block time-tagged 10 seconds prior to liftoff indication, Sense Switch 2 must be down, and all keys must be up.
 - 3) To print all information time-tagged within a certain time interval, Sense Switch 2 must be down, and the desired starting time in octal half-seconds must be entered into the keys, right justified. When MXPOCL has positioned the tape, an HTR 61348 in 61338 occurs. The ending time, also in octal half-seconds, must be entered into the keys, right justified. Press START to generate an output for the desired time interval. When this is completed, an HTR 57448 in 63008 occurs. A new starting time, which must be greater than the last ending time, now may be entered into the keys, and the process is repeated upon pressing START. If an ending time is given which is not greater than its starting time, MXPOCL prints the entire tape following the given starting time. A summary of the various stops is as follows:

```
6357_8 - Final Stop, B6 End-of-File
06356_{8}
          HTR
                  6142<sub>8</sub> - Tape positioned. Enter ending time,
061418
                             press start
                  5744<sub>8</sub> - End of time interval. Set keys, press
06325
          HTR
                             start to do another.
05732
                  5702<sub>8</sub> - B6 is unreadable
          HTR
                  5702<sub>8</sub> - B6 is a BCD tape
057378
          HTR
062128
          HTR
                             End-of-tape on B6
06345
          HTR
060738
          HTR
                             Impassable redundancy on B6. Press
061718
          HTR
                             start to skip that record and go on.
064008
          HTR
062038
                  5744_8 - B6 End-of-file while positioning tape
          HTR
065258
          HTR
                  6477<sub>8</sub> -
                             End-of-tape on A6. Put up new one,
                             press start
                  6477<sub>8</sub> - Output (A6) redundancy. Press start
06535<sub>8</sub>
          HTR
                             to rewrite
```

Most stops are accompanied by on-line printouts.

The liftoff indication is the logged entry of Mercury on-line message 219₁₀ (333₈), stating that liftoff has been received. MXPOCL does not actually search the high speed input for the liftoff bit. One may use MXPOCL to find the first occurrence of any message by replacing D340 OCT 333000000 (at alter number 556) with a constant containing the desired message number in the decrement. Similarly, the value (10 seconds) by which the program backs up after finding the message number may be varied by changing the constant A120 OCT 2000 (alter number 552) to a value containing the desired number of half-seconds, displaced 3 octal digits to the left of the low order of word A120.

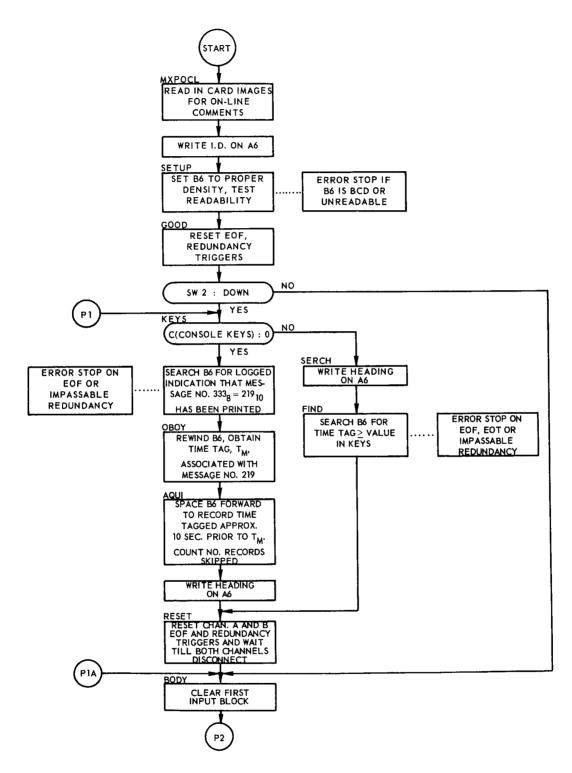


FIGURE 3-2. MXPOCL PROGRAM FLOW CHART (Sheet 1 of 6)

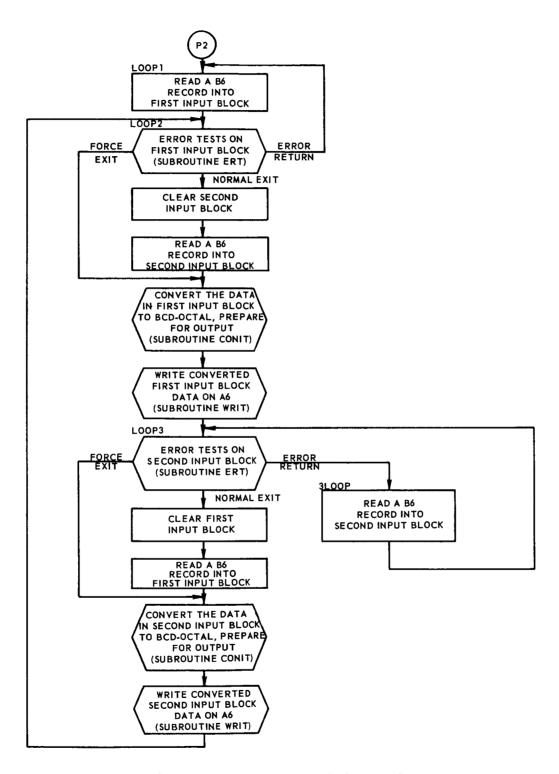


FIGURE 3-2. MXPOCL PROGRAM FLOW CHART (Sheet 2 of 6)

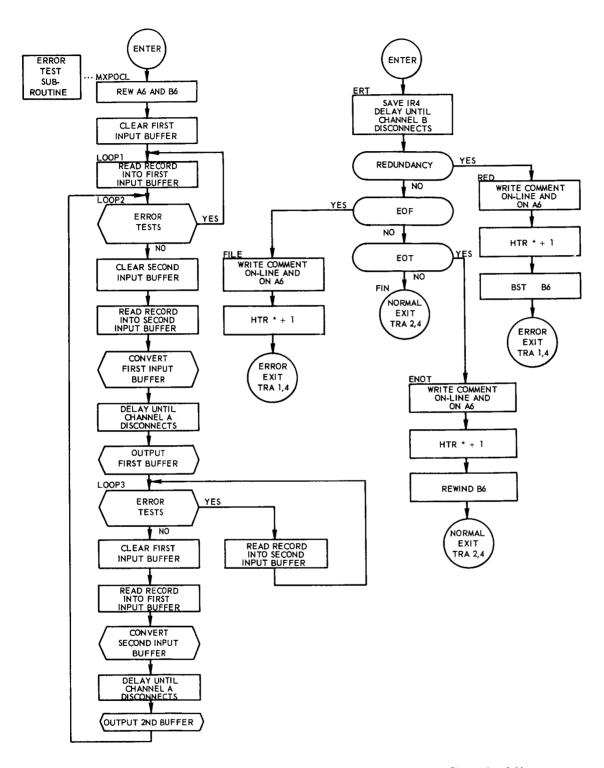


FIGURE 3-2. MXPOCL PROGRAM FLOW CHART (Sheet 3 of 6)

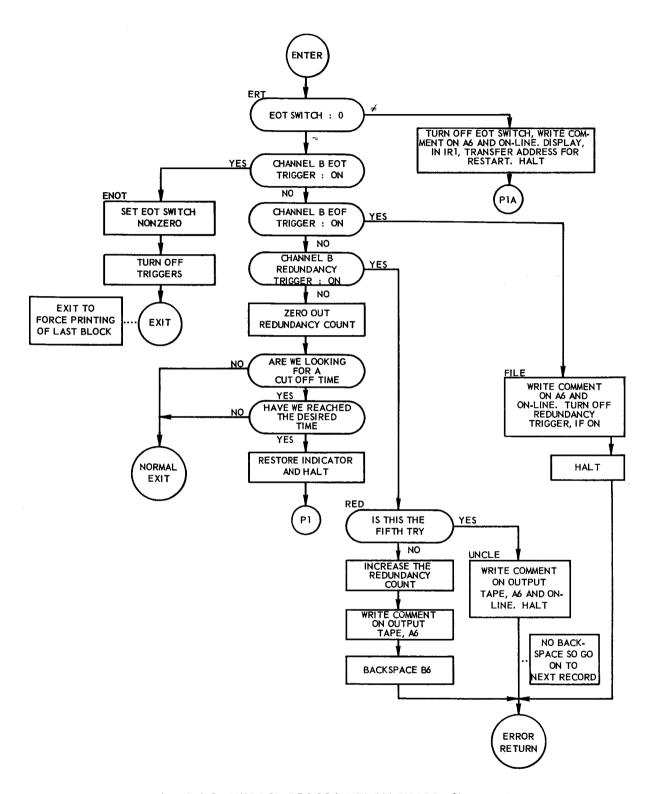


FIGURE 3-2. MXPOCL PROGRAM FLOW CHART (Sheet 4 of 6)

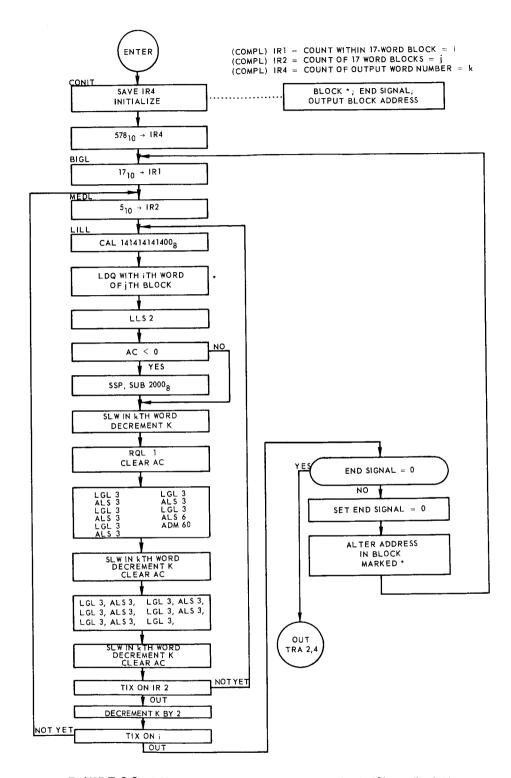


FIGURE 3-2. MXPOCL PROGRAM FLOW CHART (Sheet 5 of 6)

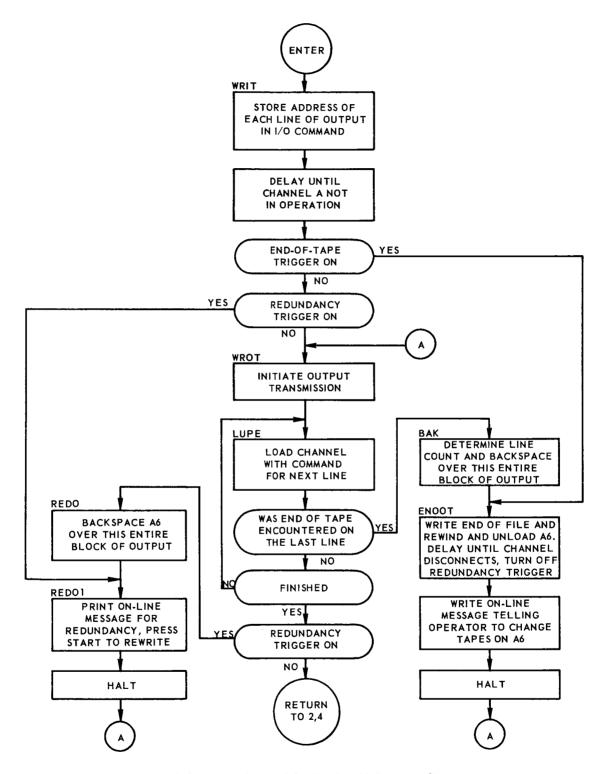


FIGURE 3-2. MXPOCL PROGRAM FLOW CHART (Sheet 6 of 6)

3.3 PROGRAM TO PRINT REAL TIME CORE'D OUTPUT (MXILCO)

MXILCO interprets and formats for off-line printing the real-time core'd output recorded on the B6 log tape by MTCOR and MSCORE. Panel information may or may not be included.

The flow chart for MXILCO is shown in Figure 3-3.

3.3.1 Input Requirements

Input to MXILCO includes the B6 tape produced during a Mercury run which contains information recorded (core'd) by RTCOR and MSCORE.

3.3.2 Output Requirements

Output from MXILCO is produced on A3, the tape used for off-line printing.

A heading will precede the first printout and identifies the MXILCO run. Each core'd output is identified with: 1) the symbol given to RTCOR, 2) the format code, and 3) the time tag. When panel information is included, it follows the heading. After panel information, the core'd information is printed, left to right, six words per line, with as many lines as needed.

3.3.3 Method

Each record of the B6 tape is tested for redundancy, EOF, and EOT. If an abnormal condition is detected, appropriate comments are printed on line and recorded on A3. When an EOT is detected before the EOF following the last record, information from the final record will be processed and an EOT comment will appear between core'd information in this record. When processing is completed following an EOT condition, an INVALID DATA comment will appear, followed by a program stop.

3.3.4 Usage

MXILCO is a self-loading, relocatable routine listed on cards in row binary.

- a) Storage Requirements-3408 locations, excluding BSS loader.
- b) Error Codes—Program stops at each EOF, EOT, or persistent redundancy. On-line messages indicate the cause of the stop and the action to be taken. Input redundancies are reread three times before the program stops.

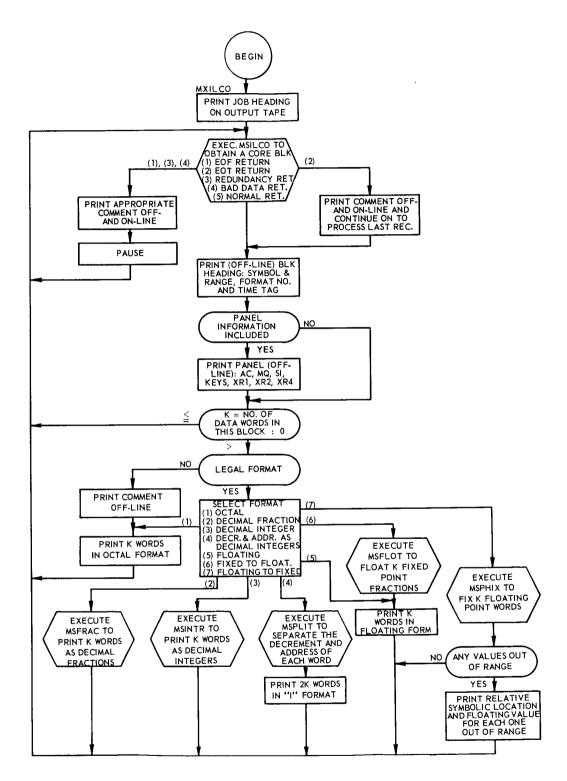


FIGURE 3-3. MXILCO PROGRAM FLOW CHART (Sheet 1 of 6)

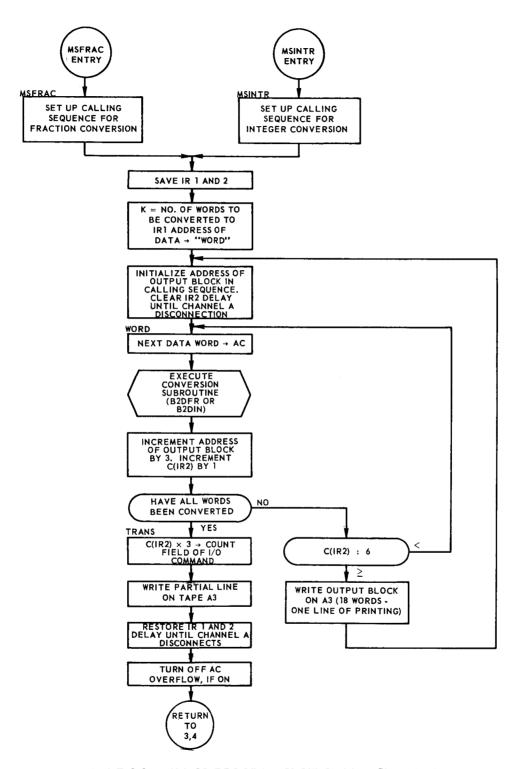


FIGURE 3-3. MXILCO PROGRAM FLOW CHART (Sheet 2 of 6)

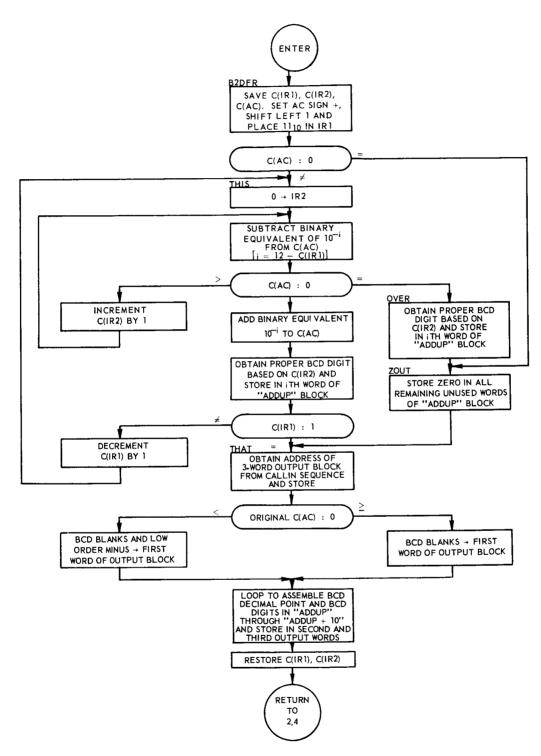


FIGURE 3-3. MXILCO PROGRAM FLOW CHART (Sheet 3 of 6)

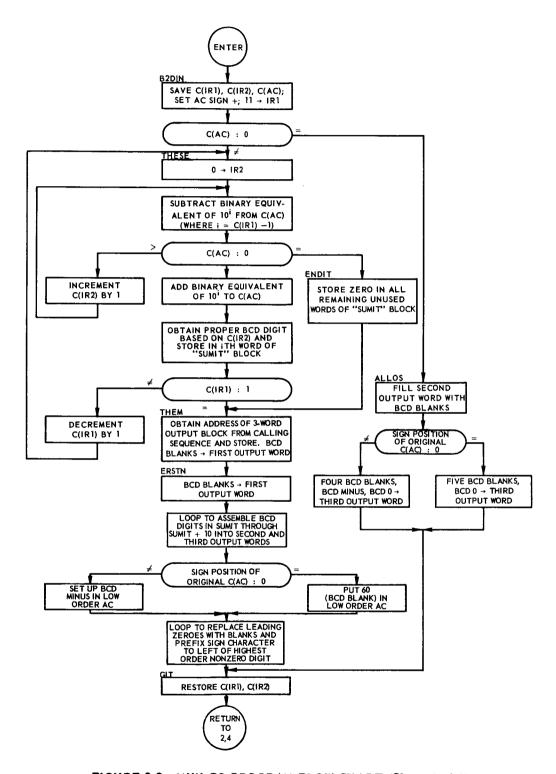


FIGURE 3-3. MXILCO PROGRAM FLOW CHART (Sheet 4 of 6)

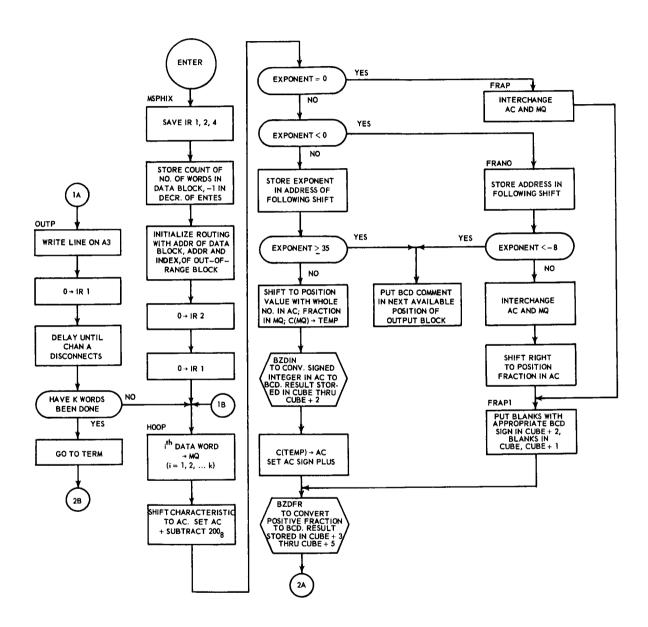


FIGURE 3-3. MXILCO PROGRAM FLOW CHART (Sheet 5 of 6)

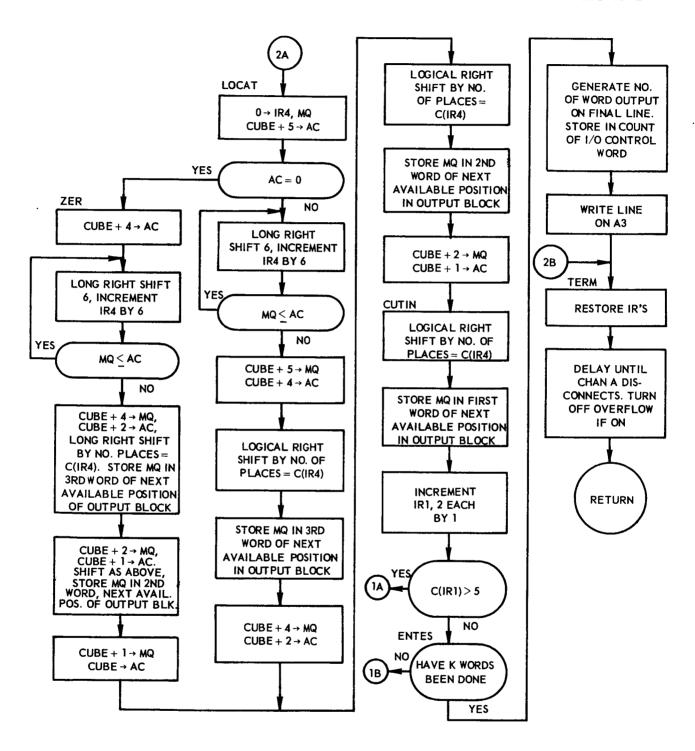


FIGURE 3-3. MXILCO PROGRAM FLOW CHART (Sheet 6 of 6)

3.4 SYMBOLIC TAPE UPDATING PROGRAM (COLSER)

COL8ER is a multipurpose utility routine used to maintain and manipulate symbolic decks in large-scale computing systems using SOS.

The COL8ER flow chart is shown in Figure 3-4.

3.4.1 Input Requirements

- a) B4 to B10 tapes (either symbolic tapes or BCD listing tapes) to be modified.
- b) B3 tape—modification packet containing:
 Special control cards (optional)
 Job card (standard SOS format and optional)
 Date card (standard SOS format and optional)
 Comments cards, columns 8-13 blank (optional)
 CPL or CPLRB (standard SOS format and optional)

These cards, if present, must be in this order.

- MACRO definition cards and associated programmer macro skeletons
- ALTER control cards and associated symbolic inserts

CHANGE control cards and symbolic inserts
GROUP control cards (see meaning below)
ORDER control cards (see meaning below)
PULMAK control cards (see meaning below)
CARD control cards (see meaning below)
GO control card

These cards or packets of cards may be in any sequence.

- c) PAUSE card must not be present.
- d) NO cards may be read on-line.

3.4.1.1 Card Definition

a) MACRO—same meaning as with SOS. The redefinition of a programmer macro with MACRO causes the old definition to be removed automatically.

depending on whether the particular COL8ER update is in the normal mode (key 1 up) or the complement mode (key 1 down). In the normal mode, ALTER has the same meaning as it does with SOS. In the complement mode, ALTER has a different definition. The cards between the numbers in the variable field of an ALTER are recorded on tape rather than deleted. For example, assume the following instructions and ALTER numbers appear on a tape to be updated by COL8ER:

300	LXA	CW, 2
301	CLA	X,2
302	SUB	Y
303	STO	${f Z}$
304	TIX	*-1,2,1

If a programmer gives ALTER 301, 303 followed by the instruction PXD 0,2 the following output results if in the complement mode:

CLA	X,2
SUB	Y
STO	${f z}$
PXD	0,2

In the complement mode, only the instructions within the numbers in the variable field of the ALTER are read out. Therefore, since the instruction of ALTER number 300 is not in the field of the ALTER, it is skipped.

In the normal mode everything is read out, except the alter cards whose numbers appear in the variable field of the ALTER card.

Normally the complement mode is used only to obtain sections of a symbolic tape. These sections can be used to create MOD(B3) tapes for subsequent COL8ER runs.

c) CHANGE—the CHANGE card is distinct from SOS. Relative numbers in the variable field count all cards, not just machine instructions. Thus, CHANGE A + 4 means "insert the following cards after the fourth card following symbolic location A, not necessarily the fourth location after location A." (Some of the intervening cards might generate no locations, such as remark cards, or might generate several locations, such as MACRO cards.)

CHANGE may refer to cards prior to the one containing the first location symbol in the program by regarding the first symbolic card in the program as number zero and by using + N as one of the parameters in the variable field of CHANGE (N means the Nth card in the symbolic deck). For example, suppose that the first few instructions of a program are:

	AXT	J, 1
	SXA	K
GO	RTBA	1
	RCHA	CW
	TCOA	*
	TEFA	*+2
	TRA	GO

To take out the instructions from GO-1 to GO+3, use CHANGE + 1, GO + 3. (Note: The variable field of a CHANGE is limited to starting and ending addresses each containing at most one symbol plus one decimal number. Example: CHANGE A; CHANGE A + 2, A + 4; CHANGE A, + 10, CHANGE + 10.)

CHANGE, when operating in the complement mode, inserts rather than deletes.

d) ORDER—the variable field contains two alter numbers; the symbolic cards between these numbers (inclusive) will be inserted on the updated symbolic tape at the point where ORDER is placed. ORDER is normally used to physically move programs or sections of programs on a symbolic tape. For example, suppose a tape contains five programs called A, B, C, D, and E. Each program has 200 alter numbers and are presently in the order of A, B, C, D and E on the tape. To rearrange the programs so that the physical order on the update tape (including one duplication) will be A, C, E, B, C, and D, the following COLSER instructions are used:

ALTER	201,1000	(Take out programs which will be reordered so that they are not duplicated on the output tape—see paragraph e below
ORDER	401,600	(Output program C after program A)
ORDER	801,1000	(Output program E next)

ORDER 201,400

ORDER 401,800 (Output programs C and D next)

e) Two important things to note in any COL8ER run is that COL8ER expects an ALTER or CHANGE to be the first instruction; if programs are being reordered and are not altered out, they appear twice on the output tape. In the above example, programs B, C, D and E (alters 201—1000) must be altered out since they are being reordered. ORDER has exactly the same meaning whether in normal or complement mode.

- f) Third Parameter (Optional)—a third parameter may be given if an INPUT card is used. This third parameter specified which deck ORDER or GROUP extracts the cards from. If this third parameter is not given, the base deck is used. Example: ORDER 5,50,4 GROUP A,B,4. Both cards refer to the fourth deck on the input tape(s). By use of this third parameter, multiple jobs may be joined in one pass.
- g) GROUP—the variable field of a GROUP card will look like that of a CHANGE card, except for the optional third parameter explained above. GROUP works exactly like ORDER, except that the variable field of a GROUP contains symbols (or, as in CHANGE, plus numbers which are relative to the first symbolic card in the programs) while ORDER contains alter numbers. GROUP has exactly the same meaning whether in normal or complement mode.
- h) PULMAK—the variable field of a PULMAK card contains the name of one programmer's macro which is not retained on the updated symbolic tape. For example, "PULMAK POLY" removes the programmer macro POLY during a COL8ER update. PULMAK is used to remove programmer macros, not to redefine them. See the MACRO instruction in this section to redefine a programmer macro.
- i) CARD-CARD control cards must appear in pairs. Any other control cards occurring between them are put onto an output mod packet for later use by SOS or COL8ER. Only control cards occurring outside pairs of CARD control cards are treated as commands to COL8ER itself and affect the selection of items to be put on the symbolic output tape.

The CARD instruction is most frequently used while running COL8ER in the complement mode. The following instructions could be used for a COL8ER run (complement mode) to create a mod tape (B3) for a subsequent COL8ER run to update a symbolic tape:

ALTER 21,28

CARD

ALTER 7,8 ALTER 20,23 CARD CHANGE A + 2, A + 9CHANGE B, B + 8AXT N, 2 CARD ORDER 300,350 ORDER 610,640 ALTER 17,18 **CARD**

The output on this run would be:

ALTER 7,8 ALTER 20,23

Instructions brought in by ALTER 21,28

Instructions brought in by CHANGE A+2, A+9 Instructions brought in by CHANGE B, B+8

AXT N, 2
ORDER 300, 350
ORDER 610, 640
ALTER 17, 18

Note that in the complement mode, if an ALTER or a CHANGE card is followed immediately by a CARD, those cards up to the next CARD precede cards brought in by the ALTER or CHANGE card.

3.4.1.2 Special Control Cards

a) INPUT—the address of the INPUT card refers to the number of jobs on each input tape in the following manner. Assume the user has six input decks: two on the first tape, one full deck and the start of another on the second tape, the balance of the deck and another full deck on the third tape, and one job on the fourth tape. The INPUT card address becomes: 1, 2, 3, 4, 4, 5, 6, 6. The first two numbers state that the tape on B4 begins with deck 1, and that deck 2 is the final deck. The

next two numbers indicate that the tape on B5 begins with deck 3 and deck 4 is the final deck. The next two numbers indicate that the rest of deck 4 is at the beginning of the tape on B6 and that deck 5 is the final deck. The final two numbers indicate that deck 6 is on B7. The INPUT card is used once for all successive jobs and is placed on the first mod deck. A new INPUT card indicating the deck allocation on the output tapes is punched when SOS is called.

- b) BASE—the address of the BASE card specifies which input deck will be updated. If the base is not specified, the deck after the last one updated becomes the base deck. Example: BASE 3 means deck 3 is updated.
- c) ONSWCH—the address of the ONSWCH card specifies which switches will be simulated as on. Example: ONSWCH 1,4 means that switches 1 and 4 are on; others off. ONSWCH 0 means that all switches are simulated as off. More than one ONSWCH card may be given.
- d) KEYS—the address of the KEYS card specifies which keys are simulated as on. Example: KEYS 40000000000 means that the sign key is on (-0 not acceptable). KEYS 122602 indicates that a date card is to be formed with this information. KEYS 0 simulates all keys off. Only one KEYS card may be given per mod deck.
- e) CONT—causes the program to begin another COL8ER pass after the present one is completed without halting between passes.
- f) SOS-causes the program to load SOS after a COL8ER pass is completed without halting after the COL8ER pass.

3.4.2 Output Requirements

- a) A3 to A9 (as many as required)—updated symbolic tape regardless of whether the input tape was symbolic or a BCD listing tape. The A3 tape may contain a SQZ pseudo-op followed by a squoze deck if these were present on the input symbolic tape and were not deleted by the mod packet. However, an input BCD listing tape may not contain a SQZ, since what follows would be the symbolic equivalent of the squoze deck, and it is the function of SOS to construct a squoze deck from the symbolic.
- b) C3 to C9 (as many as required) optional—a duplicate of A3.
- c) Punched Cards—deck of temporary modifications (optional).
- d) A 10—tape of temporary modifications (optional, but may not be selected concurrently with punched card output).

3.4.3 Method

COL8ER can perform any of the following functions:

- a) Update a symbolic tape for input to the SOS compiler.
- b) Create a symbolic tape from the BCD listing tape (A2) of a squoze deck.
- c) Create a modification packet (on cards or tape) which may be used either as input to COL8ER for modification of another symbolic tape, or together with an existing squoze deck as input to SOS for an execution run.
- d) Combine multiple decks in one pass.

Updating modification decks and symbolic tapes may include reordering and/or duplicating sections of existing programs as well as the conventional changes possible with SOS (insertions and deletions).

A single run of COL8ER yields an updated symbolic tape, containing routines from various programs if these routines are available on BCD tapes (or cards).

COL8ER can be modified to process a symbolic tape or a compilation listing tape in many ways. For example, it can be modified to search either of these tapes for desired symbolic information and produce output in a given format, such as listing all references to 6-letter characters, all transfers to 6-letter characters, all ORG cards, all transfers to subroutines with the 2-letter prefix "MS", etc. It can also be modified to produce storage maps (in list form) and to sum up storage requirements of various types of routines within a large system. Some of these modifications are available under a spearate name such as MXNDKT and CORMAP.

3.4.4 Usage

- a) Operator Procedure:
 - 1) Ready the on-line card reader with the absolute binary deck (with self-contained loader) of COL8ER. COL8ER is in the form of an SOS-produced absolute (as distinguished from relocatable) row-binary deck with 23 instructions per card. Alternatively, mount the C1 Utility Tape and ready the on-line card reader with the COL8ER call card.
 - 2) Ready the input tapes (B3, B4, Etc.), the output units (A3, A4, etc., as needed), the optional tapes (C3, C4, etc., as needed), and the A10 tape or card punch.

- 3) Set sense switches and MQ-keys to specify options desired, as listed under c5 below.
- 4) Press CLEAR and LOAD CARDS.
- b) Halts and Error Codes:
 - 1) Stop at 00003 means a serious error exists either in the machine, peripheral equipment unit, in COL8ER itself, or in the object program. The program will not restart, dump core, save tapes, and get off the machine.
 - 2) Stop at 00004 is the normal halt. Press START to
 - a) Go to SOS if the MQ sign key is up.
 - b) Perform another COL8ER run if the MQ sign key is down.
 - 3) Stop at 00005 indicates a minor delay, such as a full tape reel. Adjust, and press START.

c) Special Usage:

- 1) An advantage of the CARD CONTROL card: by using CARD carefully, a mod packet may be constructed as the output from COL8ER which will provide directions for joining multiple symbolics with a few runs.
- 2) Complement Mode: in the complement mode, ALTER and CHANGE incorporate rather than delete. All symbolic cards not within an ALTER or CHANGE field are deleted. Thus, if the programmer wants to extract a few routines from a large program, he will probably want to use the complement mode. If he wishes to insert the extracted routines into another program, he may follow the ALTER or CHANGE with CARD, the ALTER or CHANGE for the new program, and another CARD. Thus, the output from the first pass will be a mod packet for a second pass.

The output from a complement run will appear in the order of the symbolic cards of the original program. If rearrangement is necessary, ORDER and GROUP can be used.

With key 9 down, the symbolic definitions of all macros on the input tape are incorporated in the output from a complement mode run.

Note: In the complement mode, all mods are treated as permanent.

- 3) Temporary and Permanent Modifications: the letter "T" or a blank in column 72 of an ALTER or CHANGE card indicates that a new ALTER card will be punched and followed with the same mod packet. These mods will be used temporarily for SOS execution runs. Permanent mods, indicated by a letter P in column 72 of an ALTER or CHANGE card, will be included in the updated symbolic tape.
- 4) FIELD Run: the address of the FIELD card specifies that all mods with alter numbers lying in the range of the field will be automatically inserted in each mod packet for the COL8ER passes. The FIELD card must be the first card on the mod deck placed on A10. Key 18 is placed down to indicate a FIELD run. A JOB card must appear in the deck for each job used. No file mark is placed between jobs. The FIELD pass puts a new mod deck on B3 which is used for the successive COL8ER passes. The deck may have MXMRGE cards in it if desired.

Example:

MXMRGE Type Deck	Equivalent
FIELD 1,500	FIELD 1,500
INPUT 1,1,2,2	INPUT 1,1,2,2
CONT	CONT
ONSWCH 1	ONSWCH 1
JOB ONE	JOB ONE
LG	ALTER 315, 317
MOD	CLA SMTNG
ALTER 315, 317	SOS
CLA SMTNG	ONSWCH 1
ENDMOD	JOB TWO
GO	ALTER 600
SOS	ADD OTHR
ONSWCH 1	ALTER 15
JOB TWO	STL LSWR
LG	
MOD	
ALTER 600	

ADD OTHR

ALTER 15

STL LSWR

ENDMOD

GO

PAUSE

The deck produced on B3 by either deck is:

INPUT 1,1,2,2

CONT

ONSWCH 1

JOB ONE

ALTER 315, 317

CLA SMTNG

ALTER 15

STL LSWR

End of File

SOS

ONSWCH 1

JOB TWO

ALTER 315, 317

CLA SMTNG

ALTER 15

STL LSWR

ALTER 600

ADD OTHR

End of File

5) COL8ER Sense Switch Options

1 up Output to A3, unless SS5 overrides

down Output to A3, unless SS5 overrides, and C3

2 up Normal

down SOS monitor control cards not read out

	3	up	Normal	
		down	Duplicate of A3 printed on-line	
	4	up	Write EOF on output tapes and rewind	
		down	Leave output tapes positioned after last record	
	5	up	Output to A3	
		down	Output to card punch (SS5 should not be down if SS6 is up	
	6	up	Punch temporary mods	
		down	Temporary mods to A10	
6)	Key Or	otions ((Committed at the start of each COL8ER run)	
	sign	up	After halt at 4, press START to go to SOS	
		down	After halt at 4, press START for another COLSER run	
	1	up	Normal mode	
		down	Complement mode	
	2	up	Blank column 72 of ALTER or CHANGE indicates temporary mod	
		down	Blank column 72 of ALTER or CHANGE indicates permanent mod	
	3	up	Normal mode	
		down	All mods regarded as permanent	
	4	up	A3 output is in BCD	
		down	A3 output is in binary for off-line punch (not necessary with 1401)	
	5	up	A10 output is in BCD	
		down	Output is in binary for off-line punch (not necessary with 1401)	
	6	up	Normal mode	
		down	Duplicate B4; no B3 present	
	7	up	Normal mode	
		down	Temporary mods not read out	
	8	up	B3 and B4 rewound at end of run	
		down	B3 and B4 left at start of next file for stacked COL8ER runs	

9	up	Programmer macros not output in complement mode
	down	Programmer macros output in complement mode
10	up	No rewind of output tapes before run
	down	Rewind output tapes before run
11	up	Normal
	down	BCD records cut to 13 and no SQZ can be used for off-line punch (not necessary with 1401)
12	up	Normal
	down	CARD not recognized
13	up	Normal
	down	PULMAK not recognized
14	up	Normal
	down	GROUP not recognized
15	up	Normal
	down	ORDER not recognized
16	up	B4 not rewound before starting
	down	B4 rewound before starting
17	up	B3 not rewound before starting
	down	B3 rewound before starting
18	up	Normal
	down	FIELD card expected at beginning of mod deck which is on A10
19-23		Month or day in octal
24-29		Day or month in octal
30-35		Year less 1960 in octal

7) Coding Information—though usually written to produce input to SOS or COL8ER, where the input generally is the Project Mercury Programming System or some phase of it, COL8ER is an independent utility program. Since it is fast-loading and makes optimum use of 7094 I/O speeds, COL8ER itself occupies lower core memory and need not be cleared when its outputting is completed and an SOS or COL8ER run is to be made which will process that output. Pressing START after the normal COL8ER stop (at location 00004) will cause either SOS or COL8ER to be called in, as the sign MQ-key on the IBM 7094 console was up or down, respectively.

d) Checkout Status—by using a BCD listing tape of a squoze deck as the B4 input to COL8ER, the symbolic tape used to compile that deck may be reconstructed (on A3). However, several restrictions apply

1) Restrictions due to SOS:

- (a) Programmer macro skeletons do not appear on the SOS listing. Therefore, they must be made available to COL8ER on a B3 mod packet tape so as to reappear on the reconstructed symbolic tape.
- (b) A symbol which is never referenced appears in the dictionary at the end of an SOS listing with an asterisk next to the page number. A doubly-defined symbol is noted at the start of the SOS listing. However, if there is a doubly-defined symbol which is never referenced, it neither is listed in the dictionary nor printed at the start of the listing. Instead, two dictionary entries are created in the squoze deck. To ensure that the original and reconstructed symbolic tapes are really identical, the squoze decks should be compared to discover a situation such as this. The comparison should be performed with a program such as SUMARY. Also, the BCD listing tapes should be compared (for instructions only, not commentary) with a program such as COMPAR.

2) Restrictions within COL8ER

- (a) One of the SOS listing pseudo-operations, SPACE, may have a variable field N. Thus, "SPACE 3" would create three successive blank lines on the listing, using only one alter number. COL8ER, however, will generate N SPACE 1's, each producing a blank line, but each taking an alter number. This will cause the reconstructed symbolic tape, when compiled, to generate an incorrect number of alter numbers. This can be foreseen and prevented by adding to the B3 mod packet an ALTER deleting the SPACE N from the listing tape and reinserting it.
- (b) SOS programmer macros can use parameters to generate location symbols. However, if a parameter is used to generate a symbol for the first instruction generated by the macro, this symbol appears on the listing tape every time the macro is used, since the listing tape shows the first instruction generated by each macro. However, COL8ER attempts to attach this symbol to the location field of the MACRO card (since this is also a legitimate possibility), and SOS will, therefore, find one symbol being defined twice for one location. This is flagged as an error, although correctly compiled.

A further precaution: COL8ER depends upon the format of the SOS-produced BCD listing tape. If future changes to SOS modify this format in any detail, COL8ER must be changed to retain compatibility.

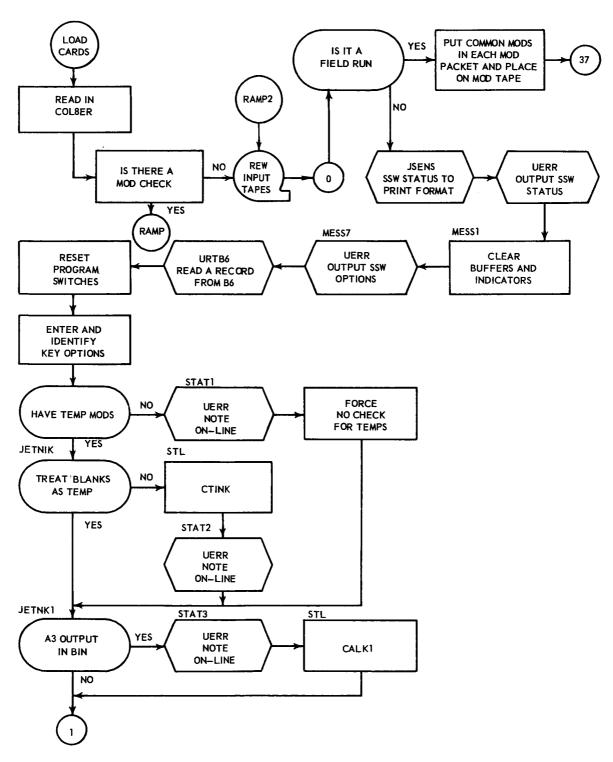


FIGURE 3-4. COLSER PROGRAM FLOW CHART (Sheet 1 of 51)

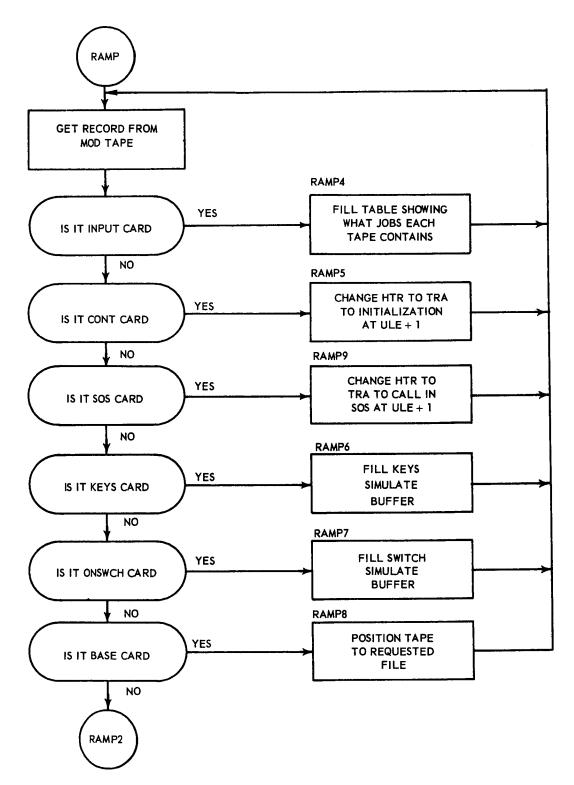


FIGURE 3-4. COLSER PROGRAM FLOW CHART (Sheet 2 of 51)

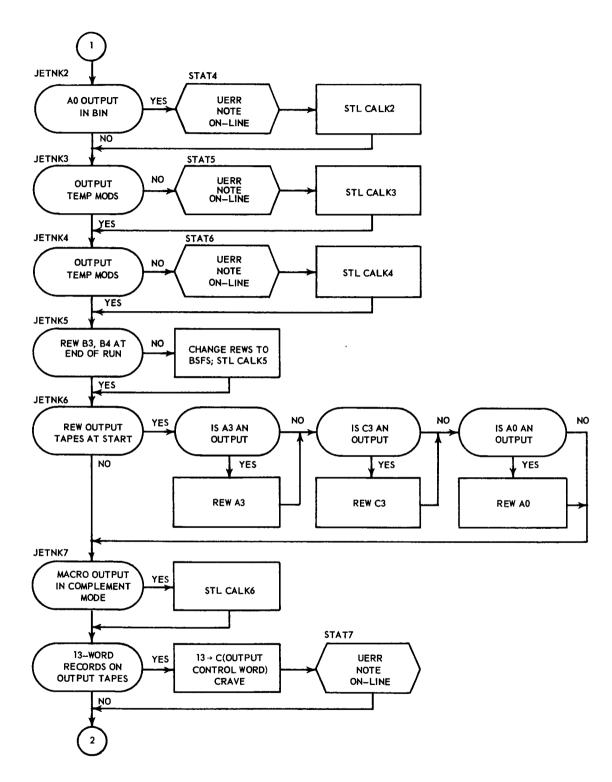


FIGURE 3-4. COLSER PROGRAM FLOW CHART (Sheet 3 of 51)

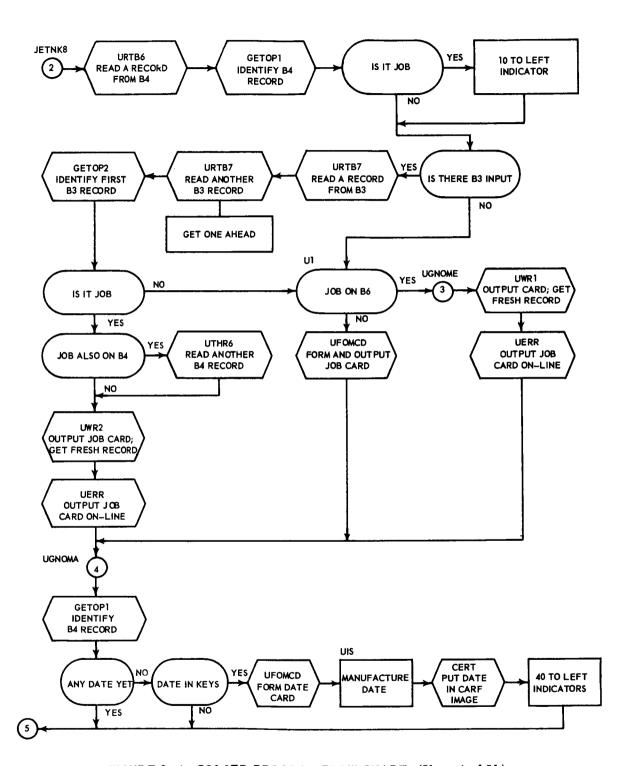


FIGURE 3-4. COLSER PROGRAM FLOW CHART (Sheet 4 of 51)

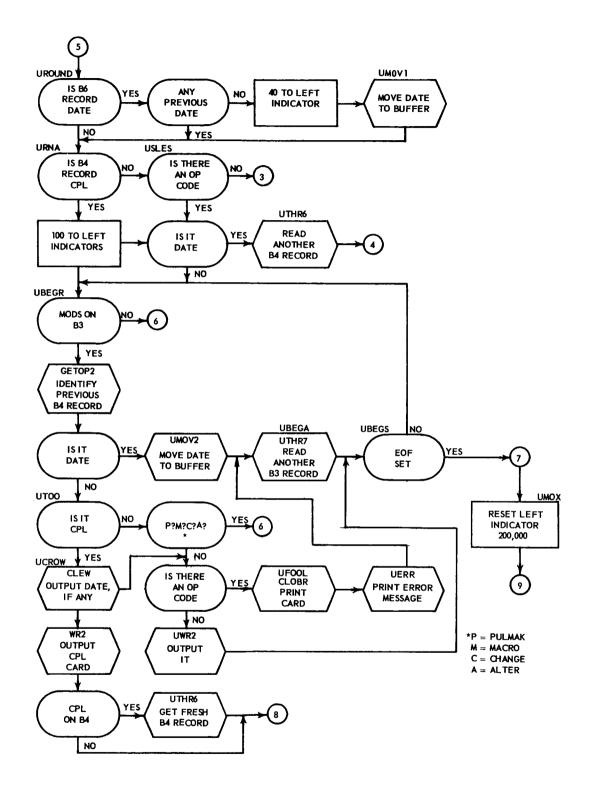


FIGURE 3-4. COLSER PROGRAM FLOW CHART (Sheet 5 of 51)

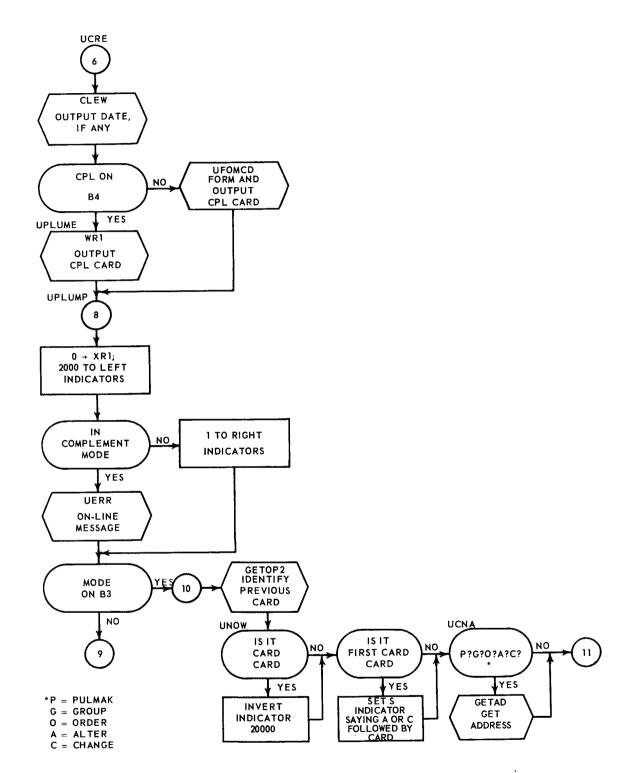


FIGURE 3-4. COLBER PROGRAM FLOW CHART (Sheet 6 of 51)

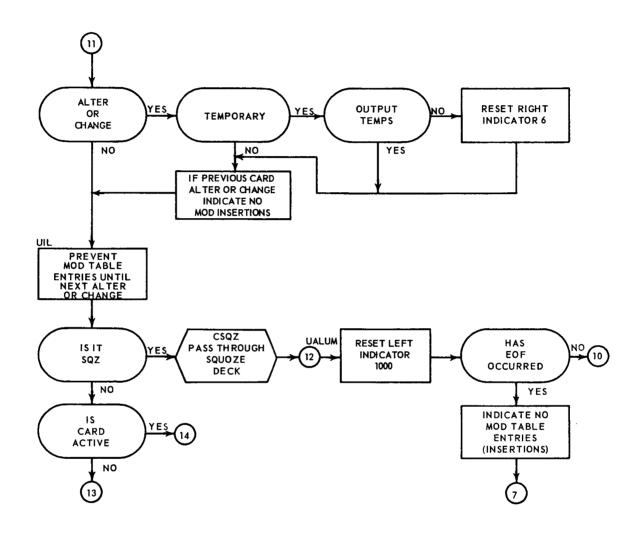


FIGURE 3-4. COL8ER PROGRAM FLOW CHART (Sheet 7 of 51)

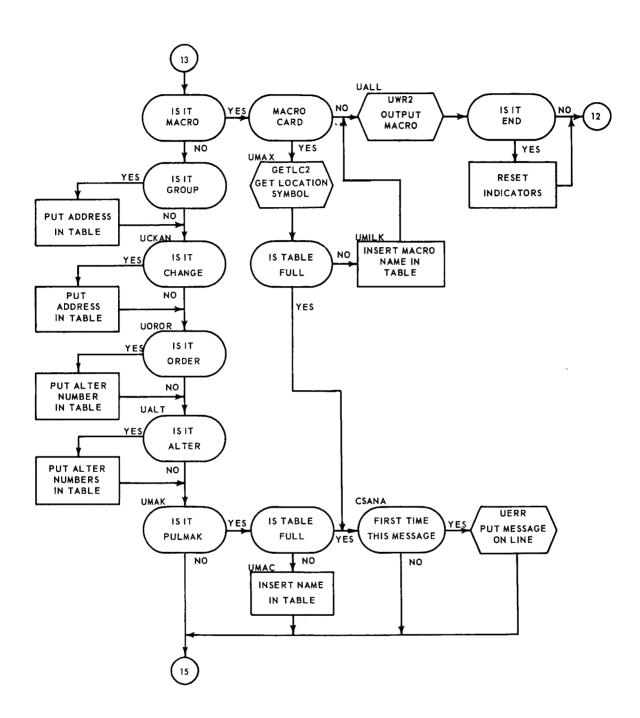


FIGURE 3-4. COLSER PROGRAM FLOW CHART (Sheet 8 of 51)

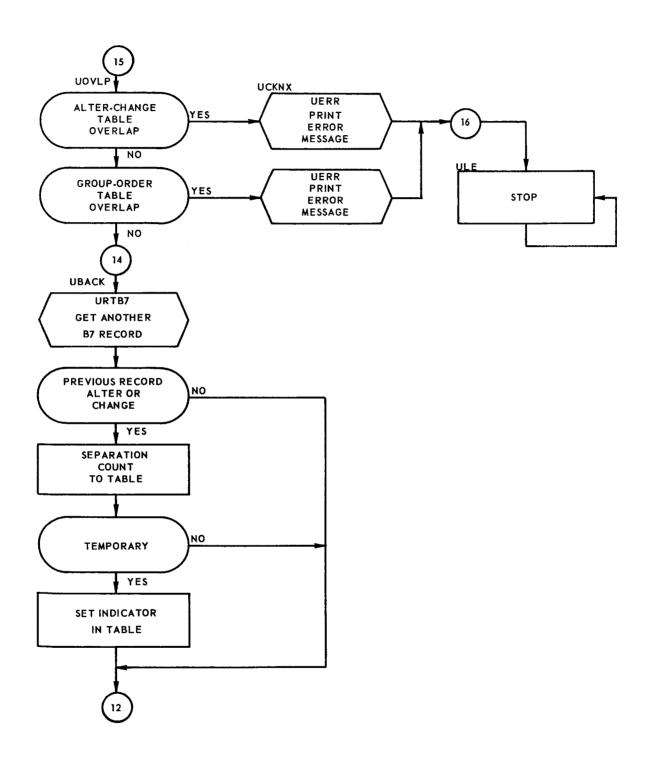


FIGURE 3-4. COLSER PROGRAM FLOW CHART (Sheet 9 of 51)

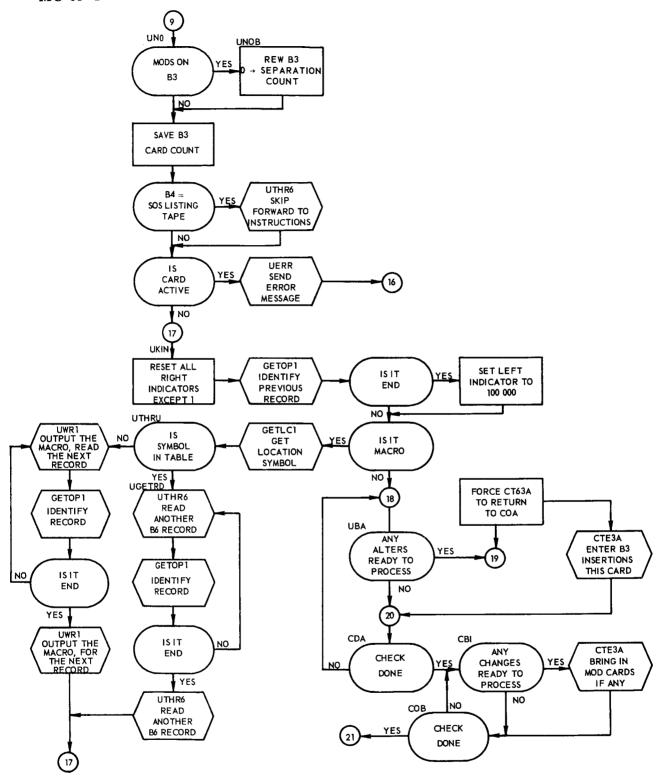


FIGURE 3-4. COLSER PROGRAM FLOW CHART (Sheet 10 of 51)

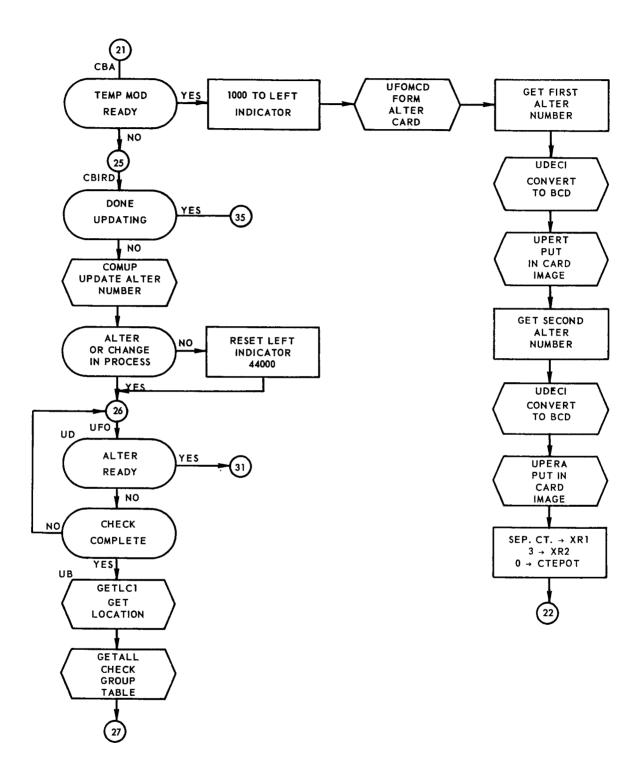


FIGURE 3-4. COLSER PROGRAM FLOW CHART (Sheet 11 of 51)

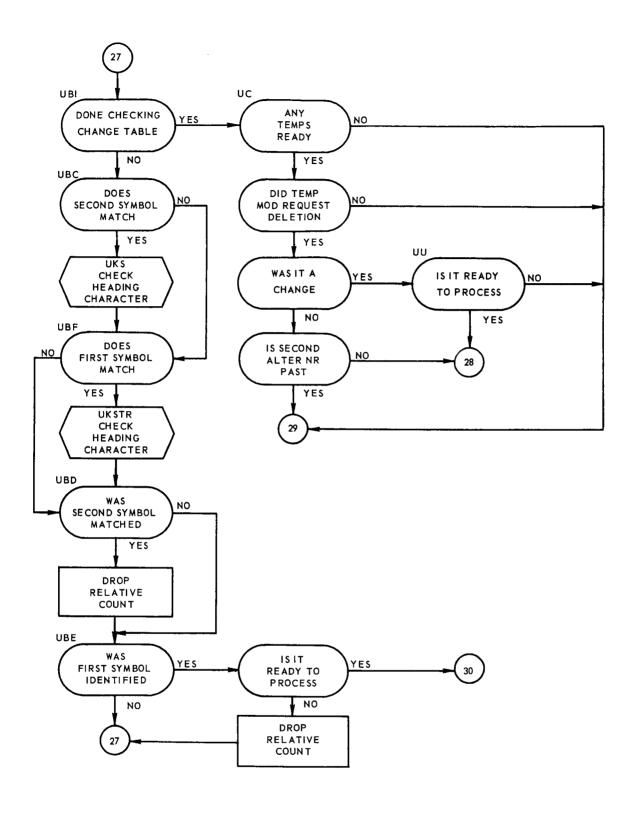


FIGURE 3-4. COLSER PROGRAM FLOW CHART (Sheet 12 of 51)

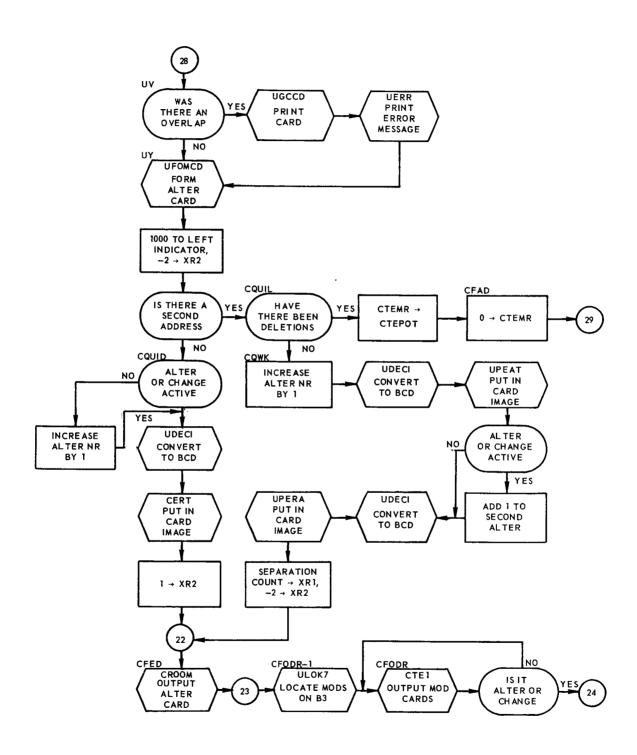


FIGURE 3-4. COL8ER PROGRAM FLOW CHART (Sheet 13 of 51)

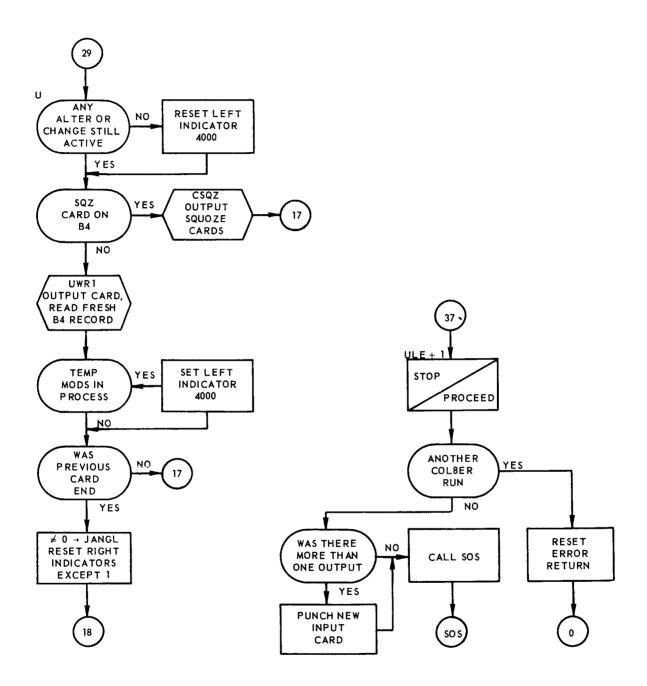


FIGURE 3-4. COL8ER PROGRAM FLOW CHART (Sheet 14 of 51)

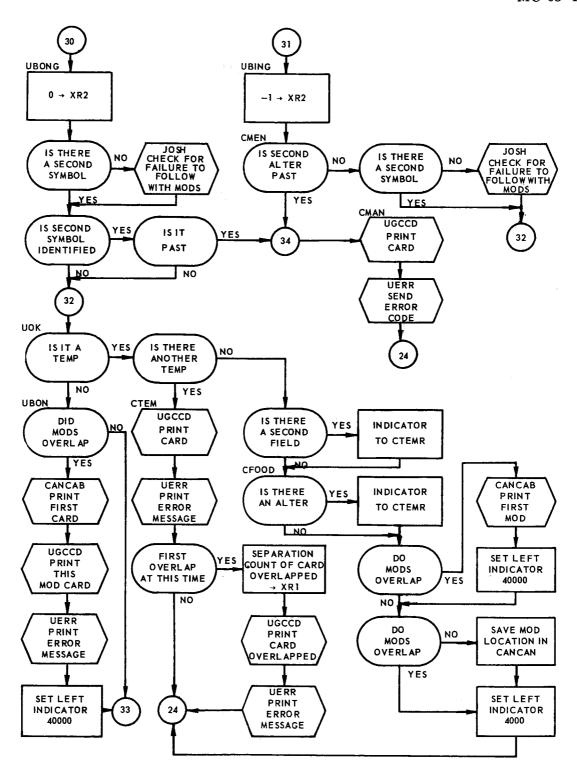


FIGURE 3-4. COL8ER PROGRAM FLOW CHART (Sheet 15 of 51)

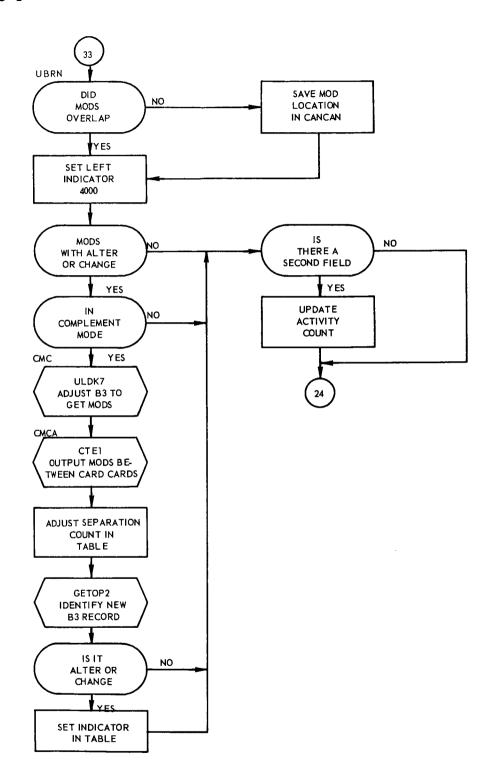


FIGURE 3-4. COLSER PROGRAM FLOW CHART (Sheet 16 of 51)

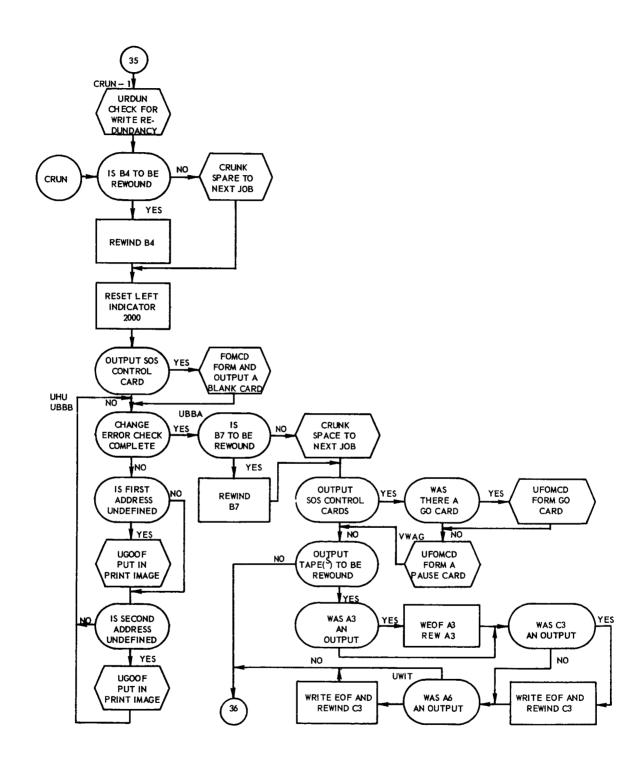


FIGURE 3-4. COLSER PROGRAM FLOW CHART (Sheet 17 of 51)

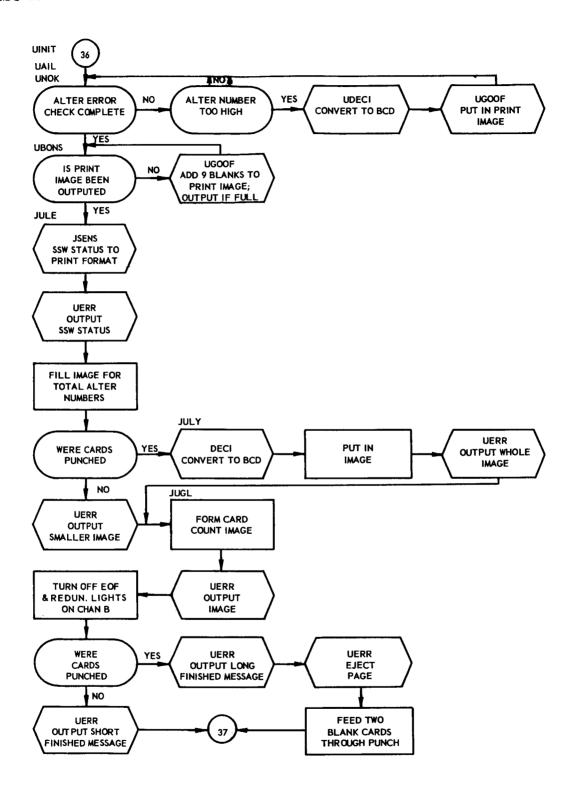


FIGURE 3-4. COLSER PROGRAM FLOW CHART (Sheet 18 of 51)

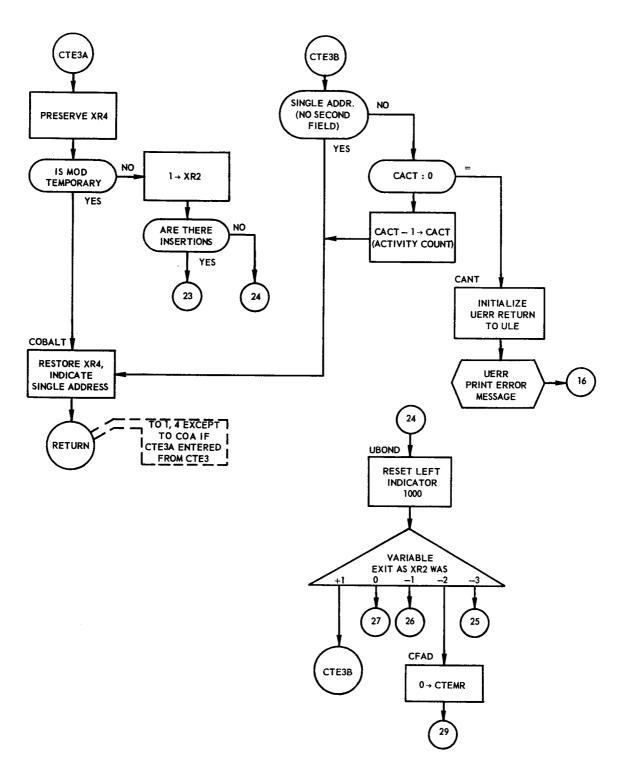


FIGURE 3-4. COLSER PROGRAM FLOW CHART (Sheet 19 of 51)

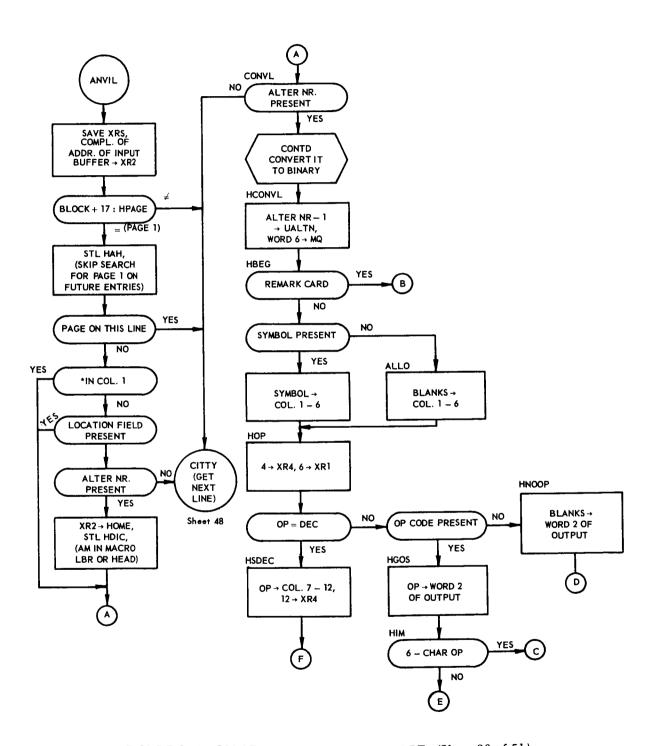
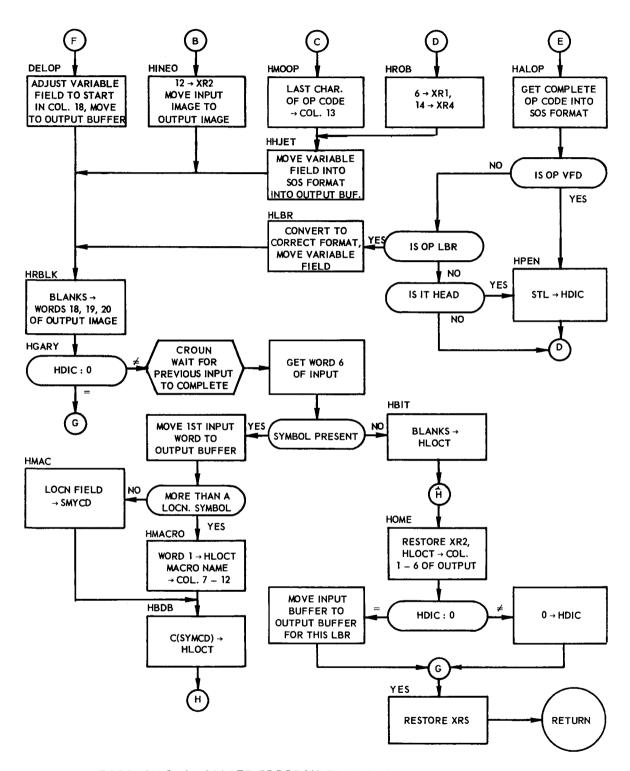


FIGURE 3-4. COLSER PROGRAM FLOW CHART (Sheet 20 of 51)



PROGRAM 3-4. COLSER PROGRAM FLOW CHART (Sheet 21 of 51)

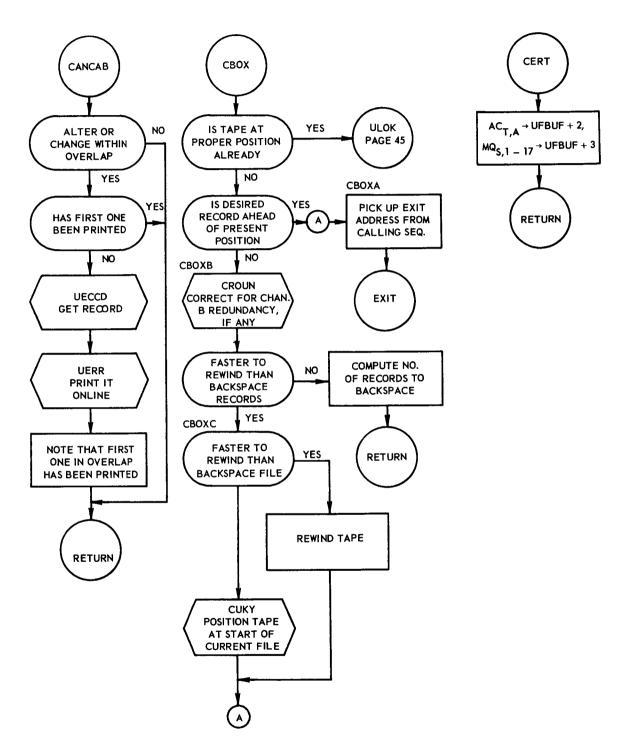


FIGURE 3-4. COLSER PROGRAM FLOW CHART (Sheet 22 of 51)

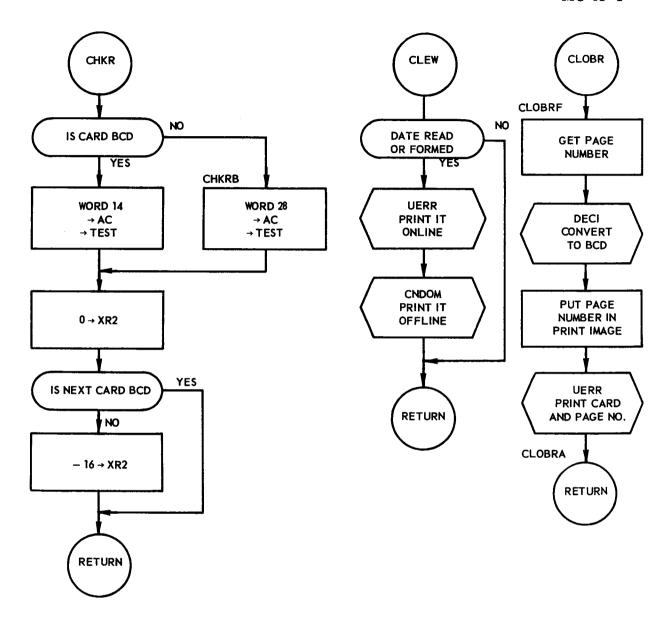


FIGURE 3-4. COLSER PROGRAM FLOW CHART (Sheet 23 of 51)

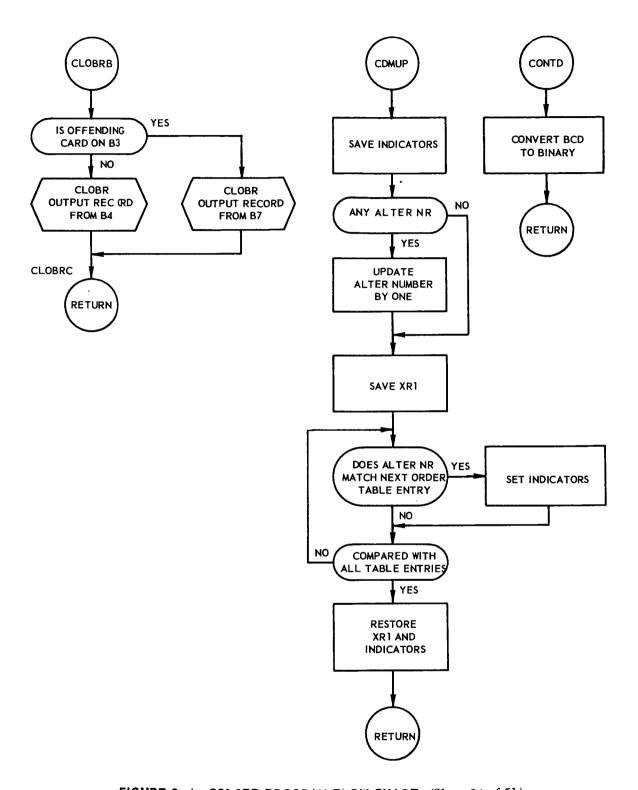


FIGURE 3-4. COLSER PROGRAM FLOW CHART (Sheet 24 of 51)

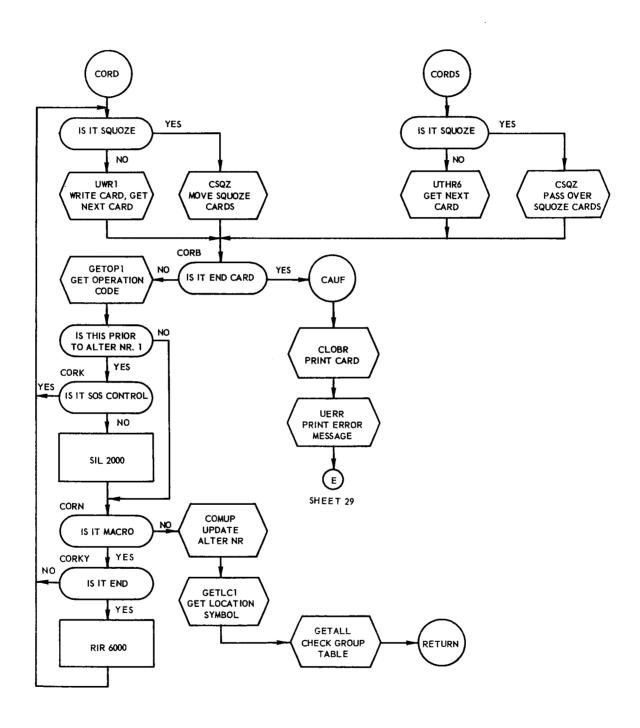


FIGURE 3-4. COLSER PROGRAM FLOW CHART (Sheet 25 of 51)

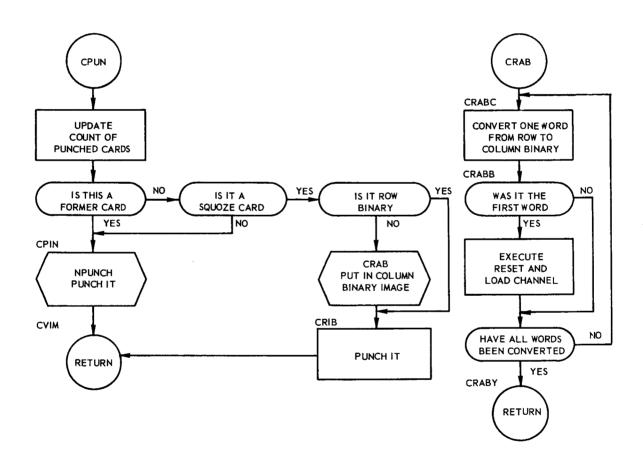


FIGURE 3-4. COLSER PROGRAM FLOW CHART (Sheet 26 of 51)

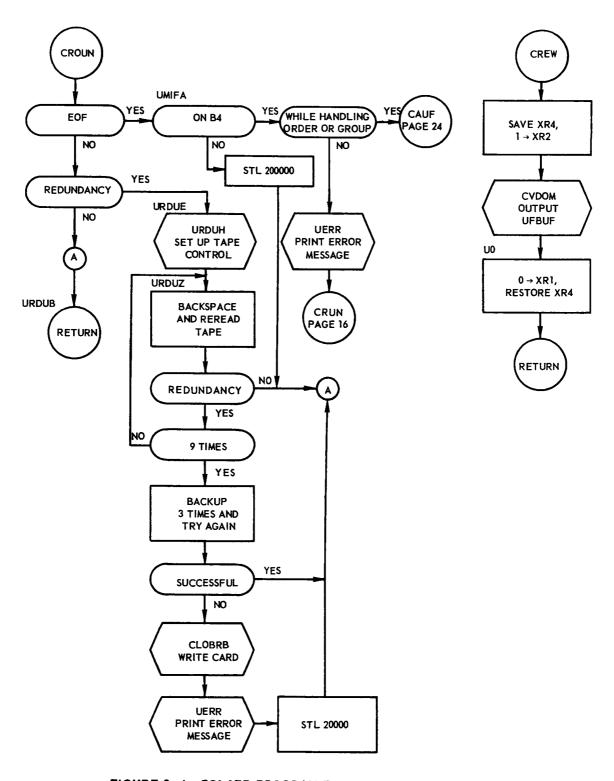


FIGURE 3-4. COLSER PROGRAM FLOW CHART (Sheet 27 of 51)

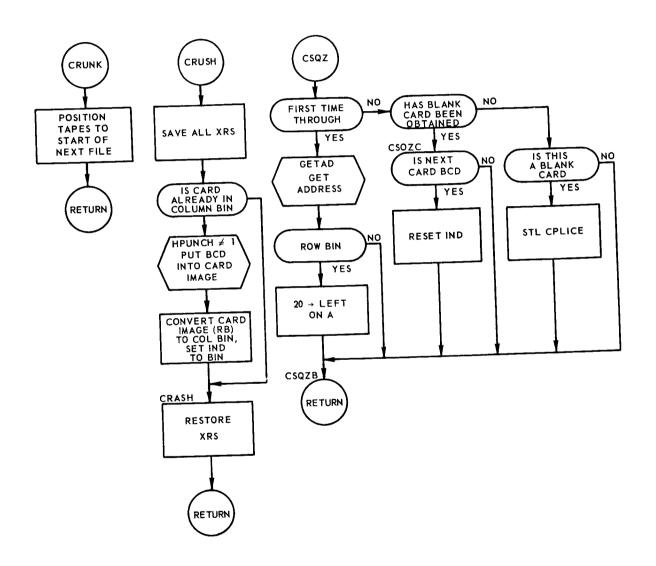


FIGURE 3-4. COL8ER PROGRAM FLOW CHART (Sheet 28 of 51)

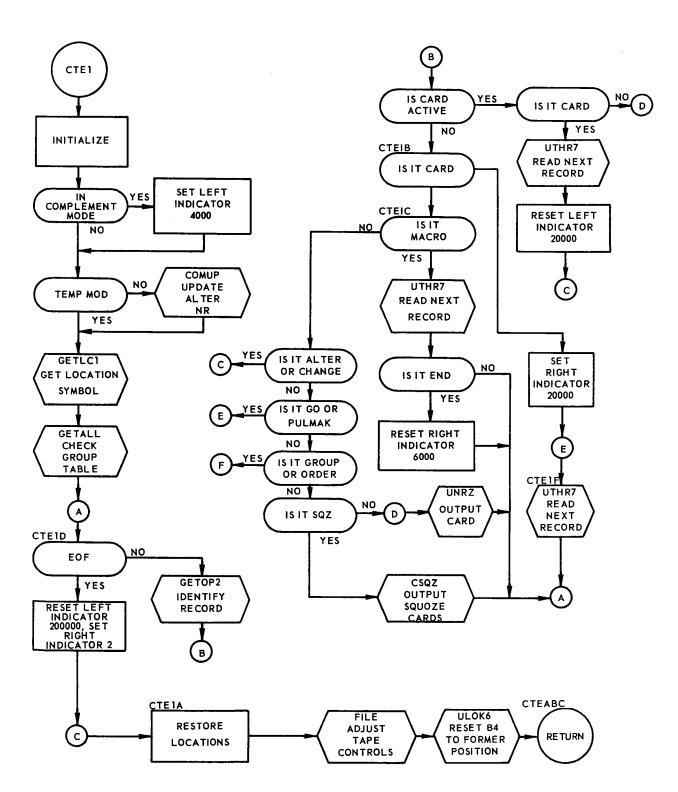


FIGURE 3-4. COLSER PROGRAM FLOW CHART (Sheet 29 of 51)

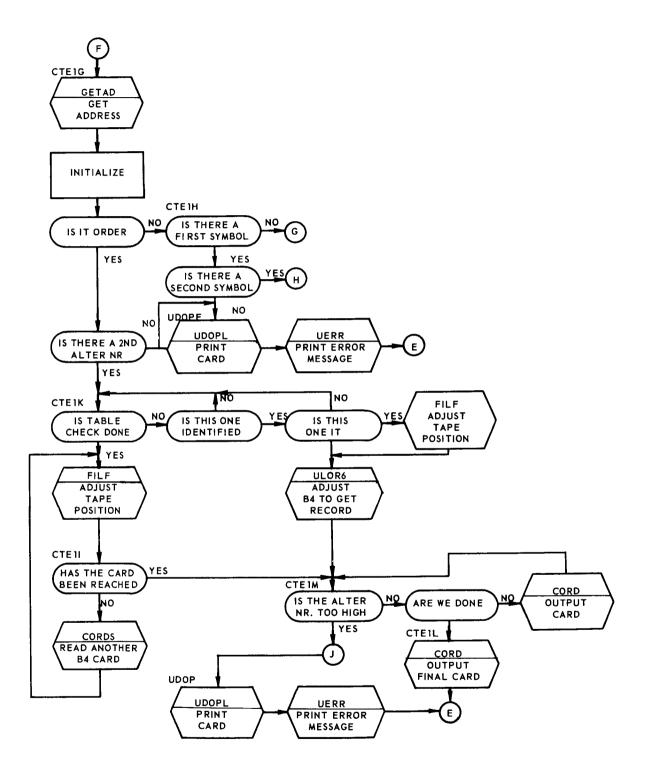


FIGURE 3-4. COLSER PROGRAM FLOW CHART (Sheet 30 of 51)

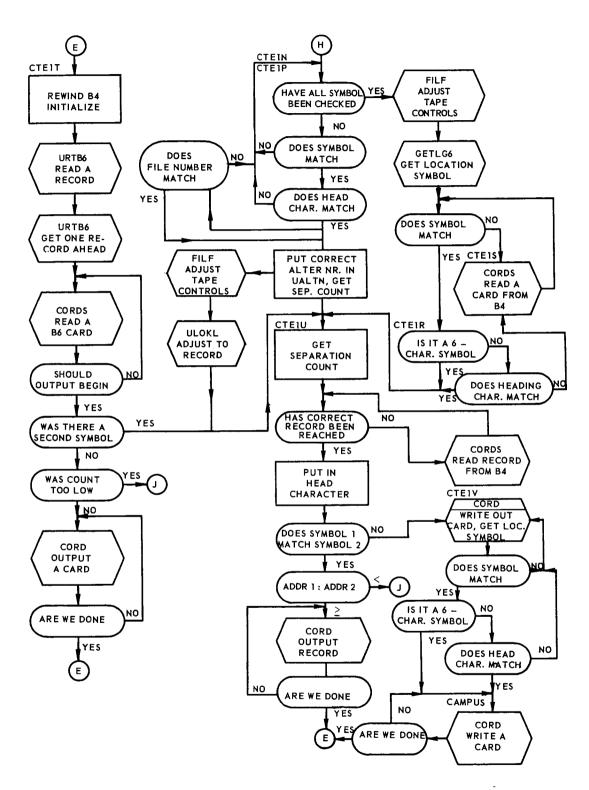


FIGURE 3 - 4. COL8ER PROGRAM FLOW CHART (Sheet 31 of 51)

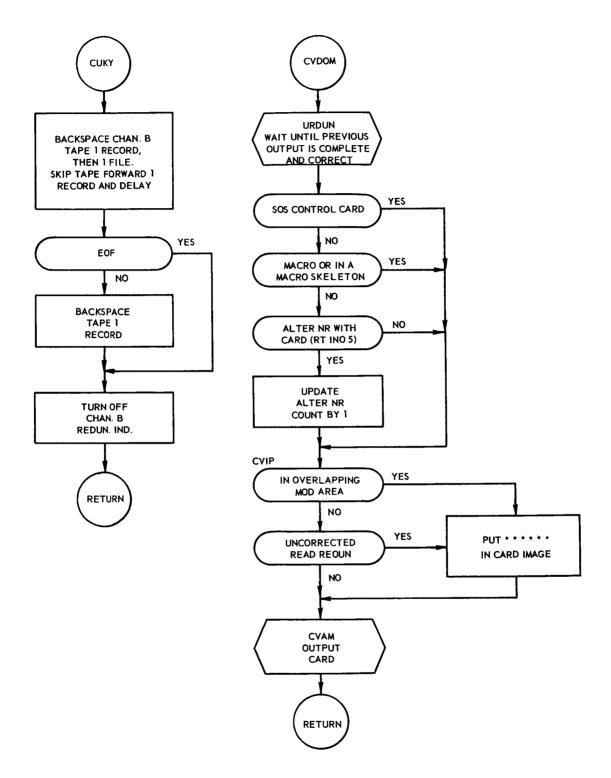


FIGURE 3-4. COLSER PROGRAM FLOW CHART (Sheet 32 of 51)

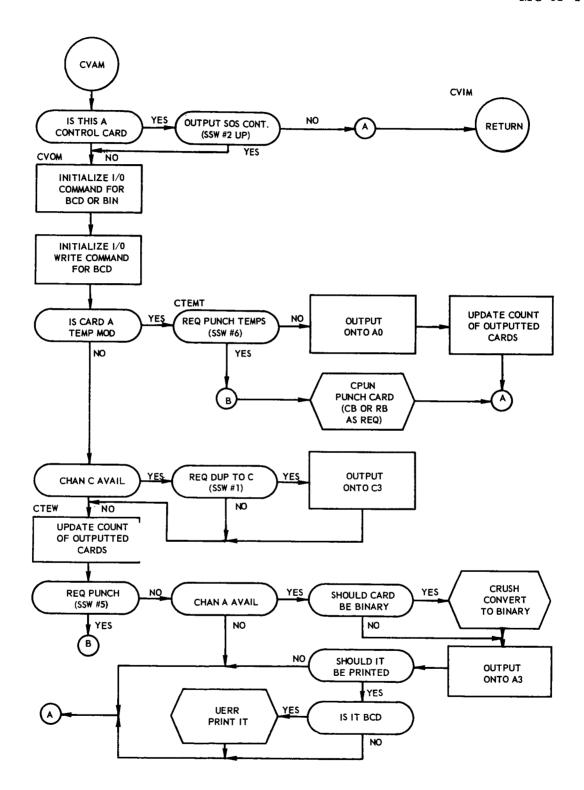


FIGURE 3-4. COLSER PROGRAM FLOW CHART (Sheet 33 of 51)

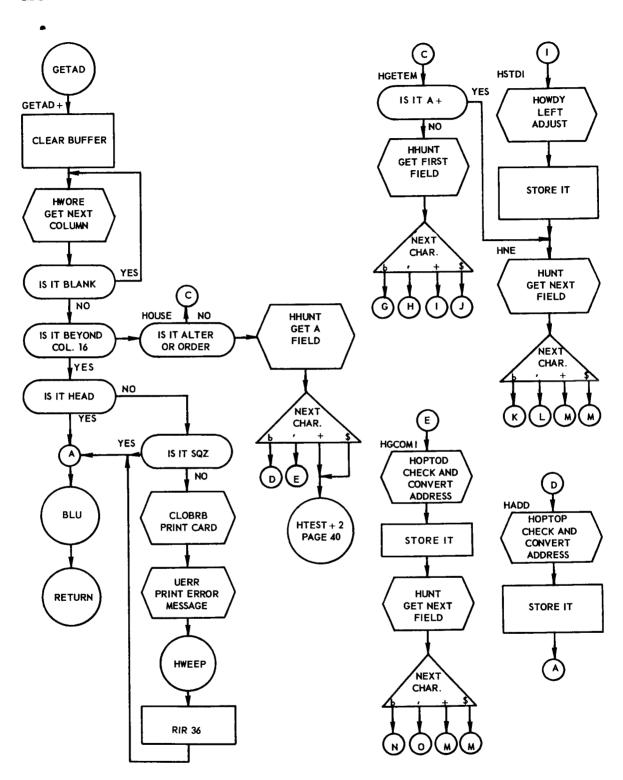


FIGURE 3-4. COLSER PROGRAM FLOW CHART (Sheet 34 of 51)

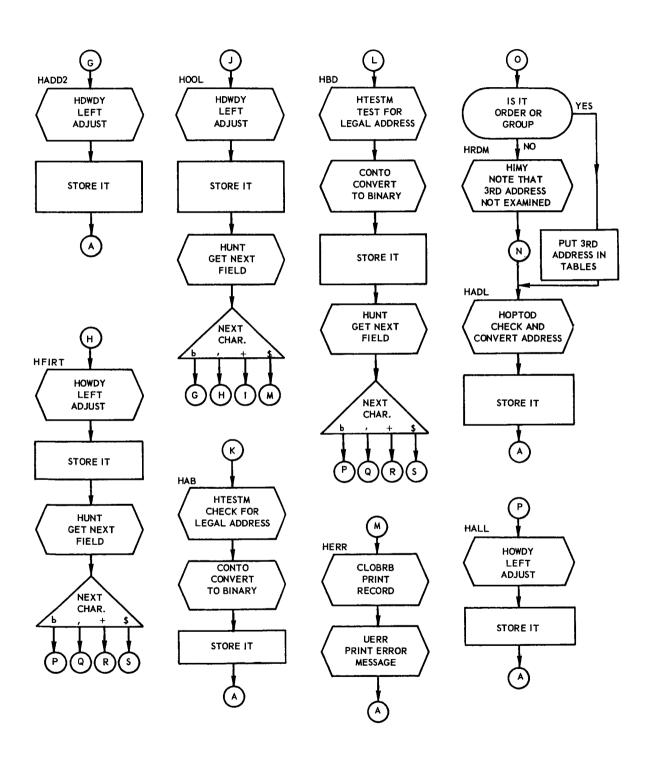


FIGURE 3-4. COLSER PROGRAM FLOW CHART (Sheet 35 of 51)

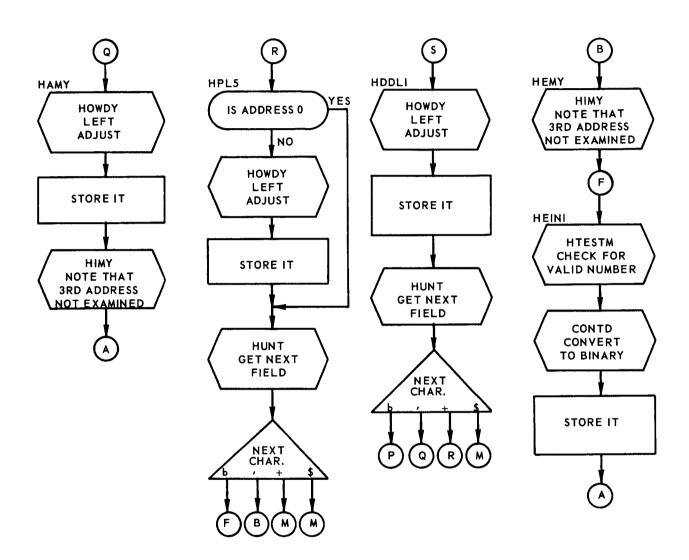


FIGURE 3-4. COLSER PROGRAM FLOW CHART (Sheet 36 of 51)

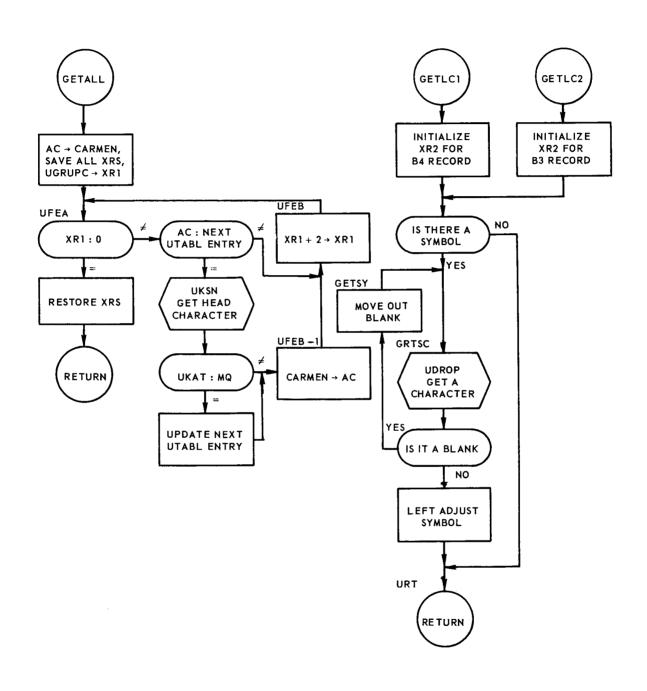


FIGURE 3-4. COLSER PROGRAM FLOW CHART (Sheet 37 of 51)

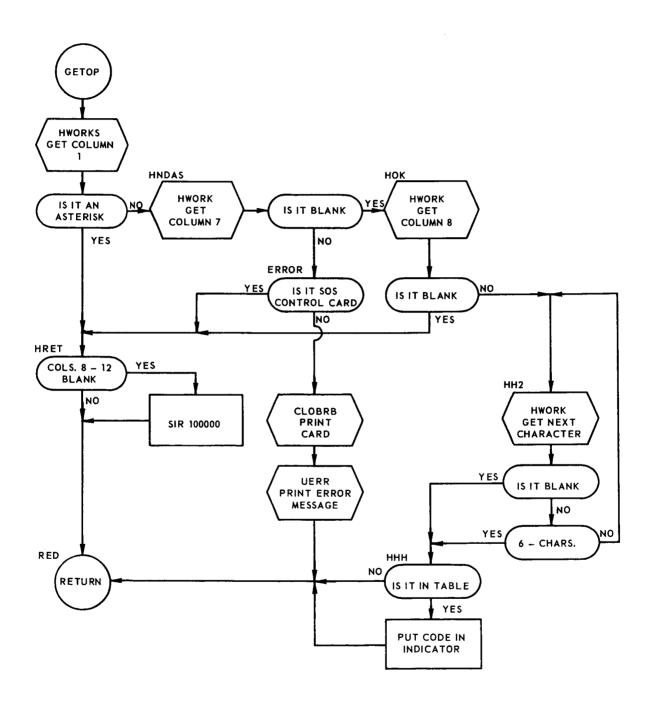


FIGURE 3-4. COLSER PROGRAM FLOW CHART (Sheet 38 of 51)

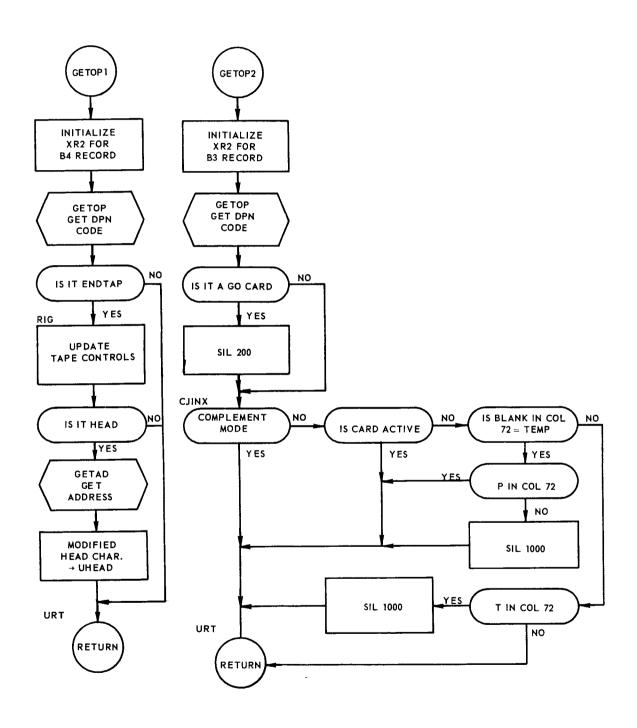


FIGURE 3.4. COLSER PROGRAM FLOW CHART (Sheet 39 of 51)

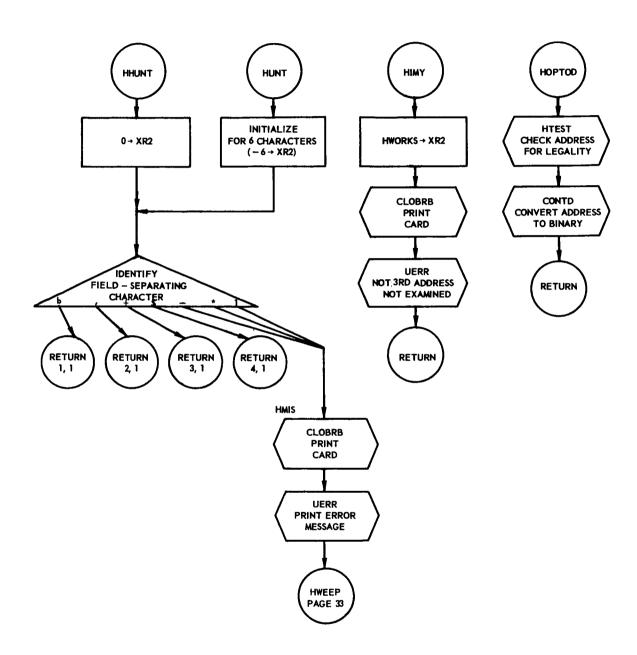


FIGURE 3-4. COLSER PROGRAM FLOW CHART (Sheet 40 of 51)

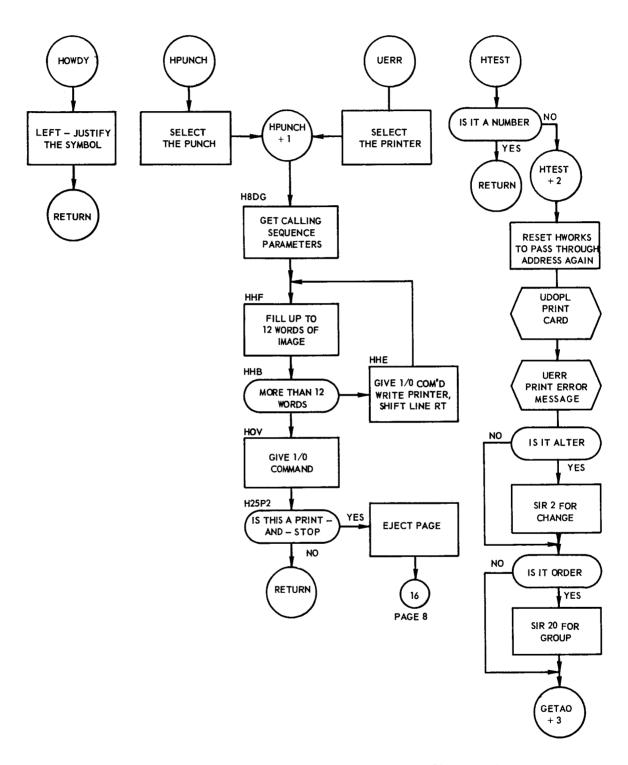


FIGURE 3-4. COL8ER PROGRAM FLOW CHART (Sheet 41 of 51)

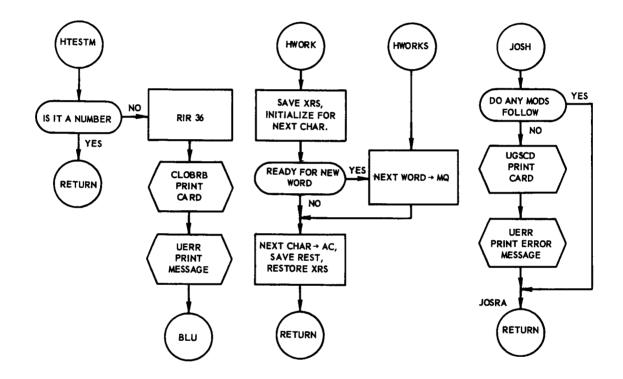


FIGURE 3-4. COLSER PROGRAM FLOW CHART (Sheet 42 of 51)

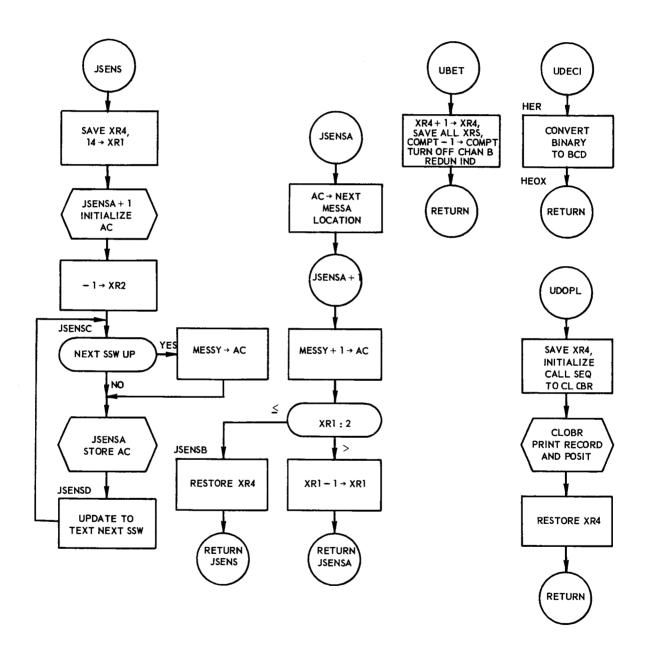


FIGURE 3-4. COLSER PROGRAM FLOW CHART (Sheet 43 of 51)

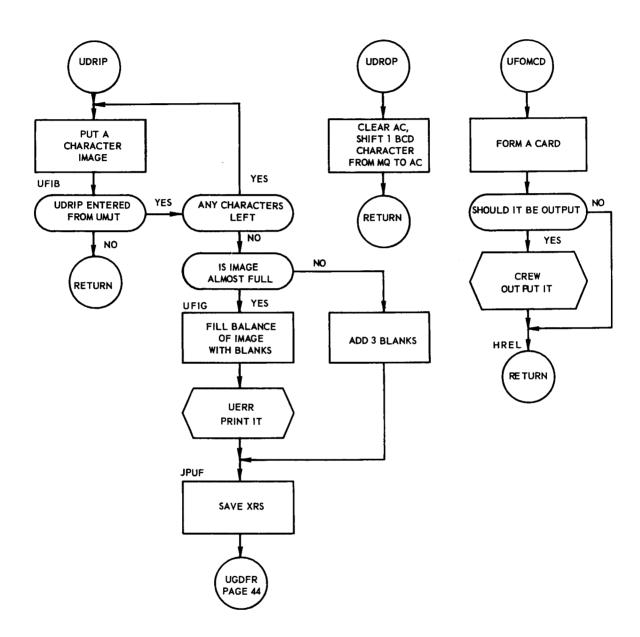


FIGURE 3-4. COLSER PROGRAM FLOW CHART (Sheet 44 of 51)

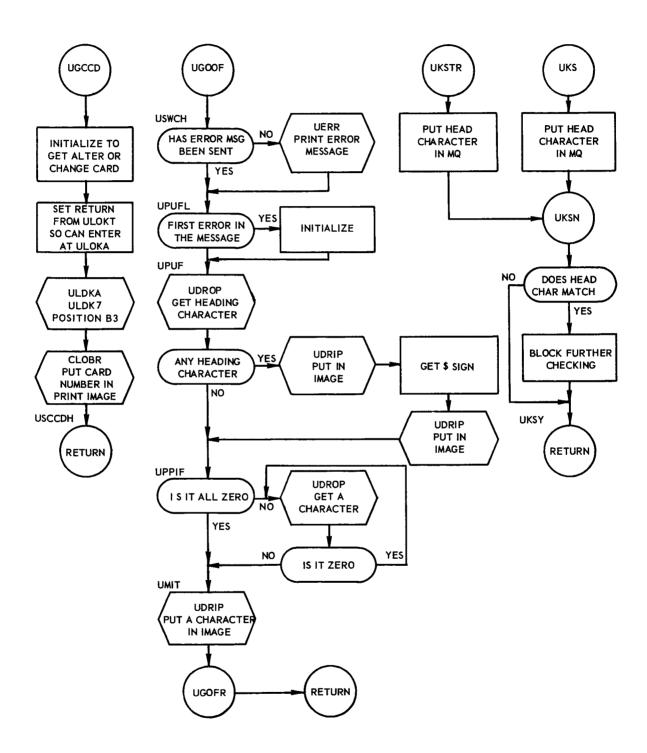


FIGURE 3-4. COLSER PROGRAM FLOW CHART (Sheet 45 of 51)

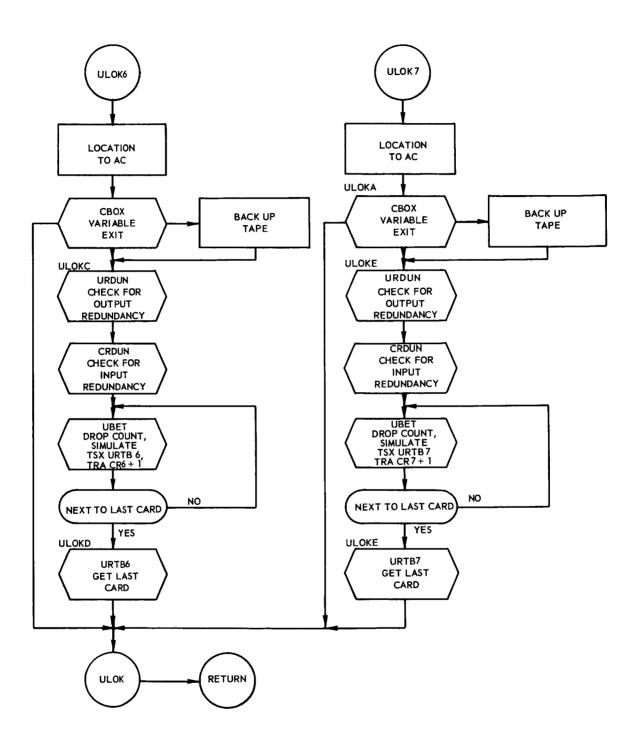


FIGURE 3 -4. COL8ER PROGRAM FLOW CHART (Sheet 46 of 51)

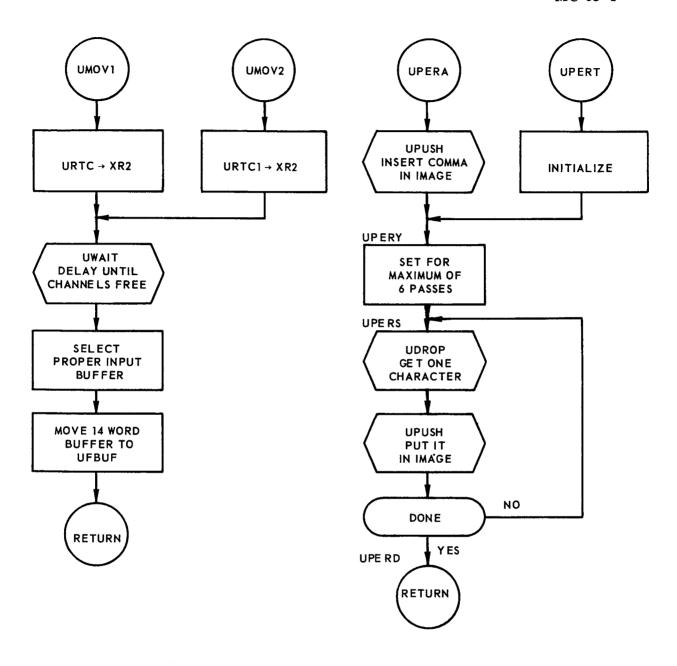


FIGURE 3-4. COLSER PROGRAM FLOW CHART (Sheet 47 of 51)

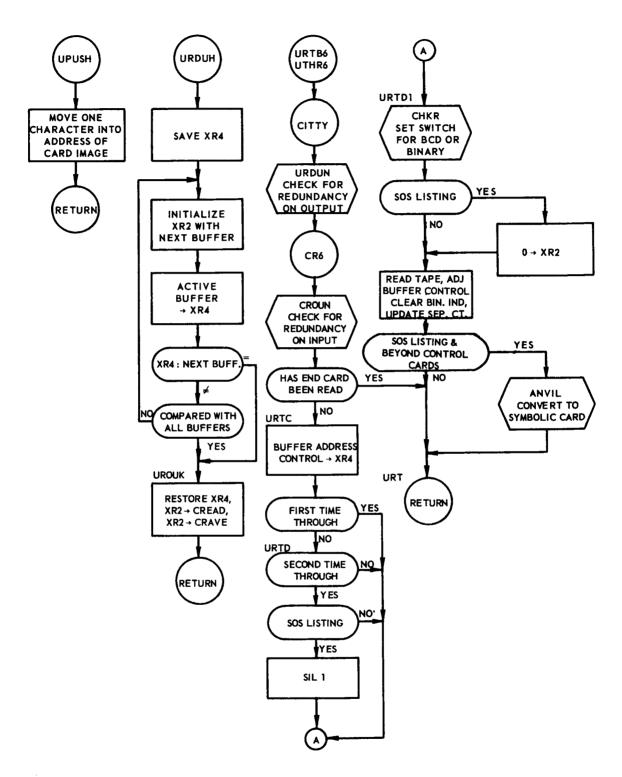


FIGURE 3-4. COLSER PROGRAM FLOW CHART (Sheet 48 of 51)

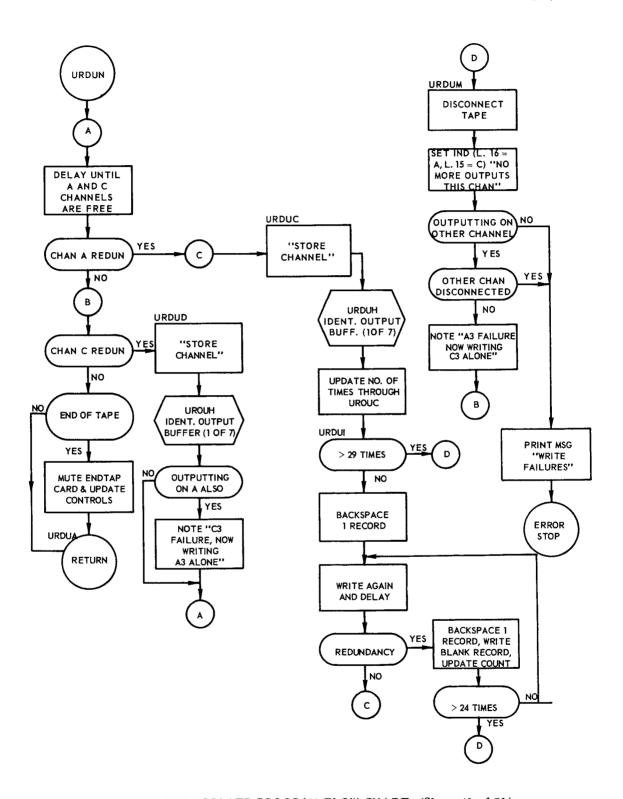


FIGURE 3-4. COLSER PROGRAM FLOW CHART (Sheet 49 of 51)

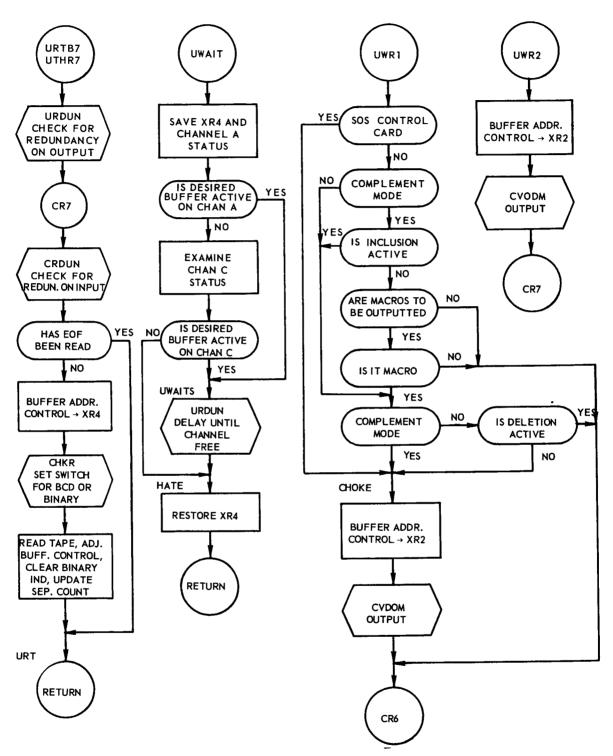


FIGURE 3-4. COL8ER PROGRAM FLOW CHART (Sheet 50 of 51)

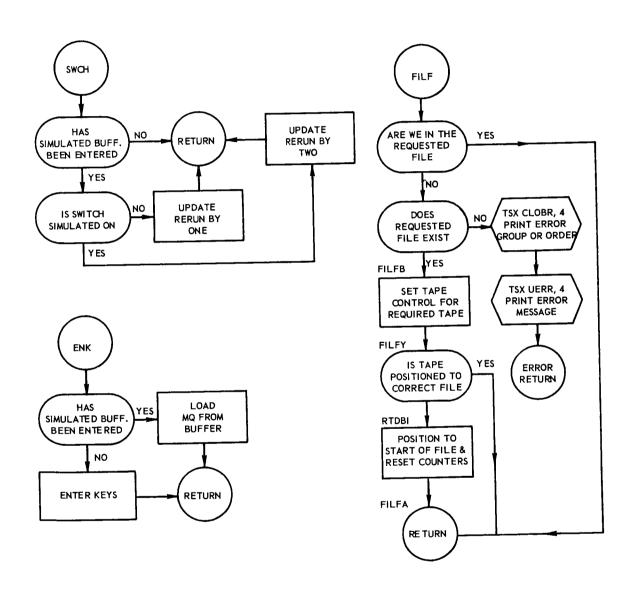


FIGURE 3-4. COLSER PROGRAM FLOW CHART (Sheet 51 of 51)

3.5 CORE MAPPING PROGRAM (CORMAP)

CORMAP records on A2 selected lines from an SOS listing. These lines contain ORG, TC, END, HEAD, JOB, USE, REFR or 6-character location fields. The last preceding line with a location field is given for ORG, TCD, or END. The following line is given with USE, REFR, or LBR.

3.5.1 Input Requirements

The BCD listing tape of the program to be examined must be placed on B6 (if it is on two reels, the second goes on B7).

3.5.2 Output Requirements

The information goes out on A2

3.5.3 Method

The tape is searched for the desired items. They are put on A2 with the decimal equivalent of the locations included. A list of section lengths beginning with 6-letter symbols is given at the end.

3.5.4 Usage

- a) Ready input BCD listing tape on B6 (and B7 if there are two tapes).
- b) Put SOS on A1.
- c) Place blank tapes on A2, B1, and B2.
- d) Place CORMAP on A3 or in card reader (sense switch 1 down if in card reader).
- e) Clear and load SOS.
- f) CORMAP halts at 5671. For additional runs, press START.

3.6 PRIORITY INDICATOR LISTING PROGRAM (MXNDKT)

MXNDKT prepares a listing of the use of "in process," "ready," and suppression indicators for the Mercury compilations. It can also list all references to 6-letter symbols.

3.6.1 Input Requirements

The symbolic tape of the program to be examined must be placed on B6.

3.6.2 Output Requirements

The indicator listing is read out on A2, and the operational 6-letter references are read out on A6.

3.6.3 Method

Every use of TRNON, TRNOF, QUEUE, and UNQUE, or their expansion along with the alter number, is placed on A2. At the end, a cross reference is made of these uses. If sense switch 6 is down, each reference to a 6-letter symbol, along with its alter number, is placed on A6.

3.6.4 Usage

- a) Ready input symbolic tape on B6.
- b) Place SOS on A1.
- c) Ready blanks on A2, B1, B2, and A6 (if sense switch 6 is down).
- d) Place MXNDKT on A3 or in the card reader (sense switch 1 down).
- e) Clear and load SOS (sense switch 6 down if 6-letter references desired).
- f) Program halts at 5670. To make extra passes, use additional GO cards and press START.

3.7 CHECKSUM CORRECTION PROGRAM (SQZSUM)

SQZSUM is a 5-card program which reproduces column-binary squoze cards with the checksum corrected.

3.7.1 Input Requirements

Column binary squoze cards suspected of having erroneous checksums serve as input to the SQZSUM program.

3.7.2 Output Requirements

This program punches on-line new squoze cards, identical to the input cards except for corrected checksums.

3.7.3 Method

The input squoze cards contain 12-bit "folded checksums" in column 3 which represent the logical sum of the bits in all data words on the card except the checksum itself. SQZSUM computes the checksum (which cannot equal zero) of each input card, even if the card itself is in error rather than its checksum, and reproduces the input card exactly, except that the checksum computed by SQZSUM replaces the previous checksum.

3.7.4 Usage

- a) Operator Procedure:
 - 1) Load and ready the on-line card reader with the SQZSUM deck followed immediately by the suspect column-binary squoze cards to be reproduced.
 - 2) Load and ready the on-line card punch.
 - 3) Press LOAD CARDS.
 - 4) Not necessary to clear memory. No sense switches or console keys are tested or used. No tapes are required. The status of all tapes is unchanged.
- b) Stops-103₈ is the correct stop (HTR 1) upon completion. Additional cards may be introduced at this time. There are no other stops.

- c) If a card in other than column binary squoze format is introduced, it is repunched without change, except for column 3.
- d) SQZSUM is a 5-card self-loading absolute row-binary deck, which loads into lower memory (using locations 0-175₈). It was assembled using SE9AP.

3.8 TAPE-KEY COMPARISON PROGRAM (KEYS)

KEYS examines a program listing tape and prints out each instruction having a direct address equal to the location entered in the console (MQ-entry) keys. Optionally, KEYS prints out each instruction having a direct address equal to any of up to 23 locations punched in a data card.

3.8.1 Input Requirements

The BCD listing tape of the program to be examined must be placed on B4 and must contain an end-of-file mark after the listing. If the EOF is missing, KEYS will examine the entire tape.

The location to be searched for is entered into the console keys in octal-equivalent BCD, e.g., location 00374 is entered as 000 000 030 704. If a data card is used, it should be placed immediately following the KEYS deck in the on-line card reader with the locations to be searched for punched, right-justified in octal-equivalent BCD, in successive words starting with the 9L word.

3.8.2 Output Requirements

Each instruction in the B4 listing with a direct address equal to the location entered in the keys or a location punched in the data card is written on the output tape, A2, in BCD. If the address field of a constant or of an immediately addressable instruction (such as AXT) is equal to a location being searched for, these constants and instructions are also written on the output tape.

No attempt is made to compensate for references to the desired location(s) via indexing or indirect addressing.

The JOB and END cards are also printed for purposes of associating the printout with the particular listing tape.

Persistently redundant records are printed out with the word REDNCY inserted at the right.

3.8.3 Method

KEYS reads in one record at a time from B4 and compares the address field with the location specified either by the setting of the entry keys or punched in the data card. If they compare, the record is written on A2 and a counter is updated. KEYS continues to read individual records and compare their address fields until an end-of-file is reached; whereupon KEYS prints out the count of its findings, rewinds the input tape, and halts. At this time, a new data card or

key setting may be entered, sense switch 1 reset, and START pressed to initiate another pass. After a pass using a data card, the keys may not be used to enter data for future passes.

An examination of location 00000 may be made with a zero entry in the keys. If a data card is used and location zero is one of the locations being searched for, the 9L word of the data card must be zero.

3.8.4 Usage

- a) Ready input BCD listing tape on B4.
- b) Ready blank tape for output on A2.
- c) Enter location in console keys in octal-equivalent BCD. Optionally depress SSW 1 and place the data card behind the KEYS deck.
- d) Ready the on-line card reader with the KEYS deck (and loader), followed by the data card(s) if more than one location is to be searched for on one pass.
- e) Press LOAD CARDS.
- f) KEYS is a 9-card absolute row-binary deck assembled by SOS. It requires a 2-card loader.
- g) Final stop, at 5673₈, is an HPR. If finished, write an end-of-file on A2, press CLEAR, and print A2 under "single space." To initiate another pass, (1) reset SSW 1, (2) enter new control card or reset keys, and (3) press START.

3.9 LOW CORE REFERENCE PROGRAM (LOWCOR)

LOWCOR examines a program listing and records each instruction with a direct address less than 0040.

The flow chart for the LOWCOR program is shown in Figure 3-5.

3.9.1 Input Requirements

The program to be examined should be listed (using LS, PS or CPL) on tape which becomes input to LOWCOR on B4. An end-of-file must be written on the listing or LOWCOR does not terminate until encountering an end-of-tape mark on B4. The program reads in and tests one record at a time.

3.9.2 Output Requirements

The selected records from the examined program are written on tape B5 for off-line printing. The program writes an end-of-file on B5 and rewinds B5 prior to returning to SOS Monitor. Constants, immediately addressed instructions (such as AXT and ALS), instructions with effective addresses which result from using the XEC instruction, indirectly addressed instructions, and the various pseudo-instructions (such as ORG, BSS, EQU, END) are written on B5 if their direct addresses are less than 0040. The JOB card and the page numbers from the input tape are reproduced on the output tape.

3.9.3 Method

The method of operation for LOWCOR is illustrated in Figure 3-5.

3.9.4 Usage

- a) Time Required—LOWCOR requires, in addition to the time needed for rewinding tapes B4 and B5 at the beginning and end of the program, approximately 5N milliseconds, where N is the number of records in the program listing being examined.
- b) Storage Required-80 core storage locations (11610 $_8$ 11727 $_8$).
- c) Checkout—LOWCOR was used to check all sections of the SOS system and was hand-checked by comparing it with the listings of sections DA and M1.

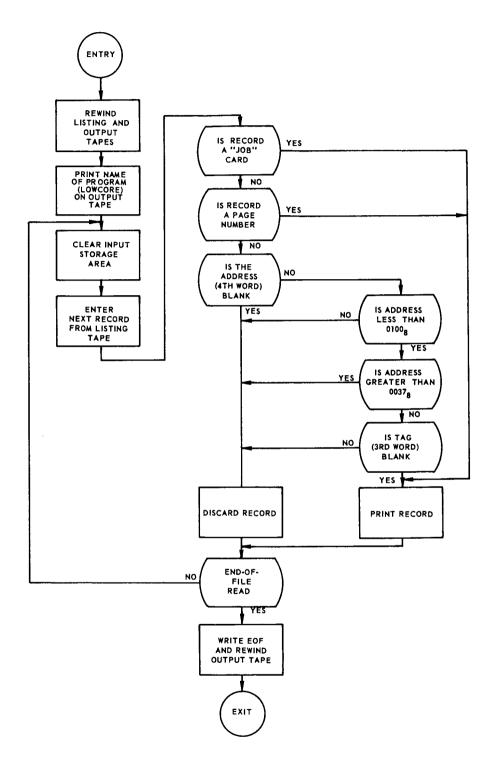


FIGURE 3-5. LOWCOR PROGRAM FLOW CHART

3.10 SQUOZE DECK COMPARISON PROGRAM (COMPAR)

COMPAR compares corresponding records from two program listings and records the locations of those which are not identical. One record at a time is read in from each listing tape and compared.

3.10.1 Input Requirements

The programs to be compared are listed (using LS, PS, or CPL) on tapes which then become the input tapes B3 and C2 for the main program. An end-of-file must follow both listings or COMPAR does not terminate until encountering an end-of-tape mark on B3 or C2.

3.10.2 Output Requirements

The locations of corresponding instructions which are not identical are read out on tape A2 for off-line printing.

3.10.3 Method

COMPAR reads in one record each from B3 and C2. If the instruction portions of the two records are not identical, the location of the record is written on the out tape. When an end-of-file is read from either input tape, both input tapes are rewound and the program stops. Commentary, pseudo-instructions, BCI variable fields, and other nonexecutable information, which may have been spaced or punctuated differently in the two program versions without affecting their execution, are ignored.

3.10.4 Usage

- a) Time Required—the time required for COMPAR (IBM 709) is 6 + 3N + D + S milliseconds, where N is the number of records in the shorter of the two programs being compared; D is the number of times the error routine, upon encountering unlike corresponding records, has to be employed; and S is the number of times the search routine has to be employed to realign the tapes.
- b) Storage Required-468 cells.
- c) Operator Procedures:
 - 1) COMPAR is a squoze deck executed under SOS control. Therefore, tapes A1 (SOS), A2, B1, and B2 (blanks) are required. The

- COMPAR deck should be readied in the on-line card reader and sense switch 1 depressed.
- 2) During execution, upon an HPR after an on-line message, the tape numbers of B3 and C2 should be entered in the keys and sense switches. The right three digits of B3 should be entered in BCD in the decrement of the keys and of C2 in the address. The fourth (high order) digit is entered in the sense switches. For example, B3 = H1579, C2 = H81 would be entered as:

keys: 050711001001. sense switches: 001000.

- 3) Operate as a normal SOS job.
- d) Restriction—information not available on a listing tape, such as programmer macro expansions, LBR cards, or expansion of other generative pseudo-instructions, cannot be compared for correctness.

3.11 SYMBOLIC CORE DUMP PROGRAM (MXCORE)

MXCORE is a self-loading program which causes a symbolic rescue dump of core after an SOS program has destroyed the SOS Monitor. MXCORE dumps core memory, exclusive of SOS Monitor, onto B2 in the SOS format and restores SOS while producing the output tape for printing on A2.

The flow chart for MXCORE is shown in Figure 3-6.

3.11.1 Input Requirements

MXCORE, as a salvage measure following a program mishap, requires that the SOS tape setup be untouched—no tapes may be moved. The RESET button should not be used and the CLEAR button must not be used. Sense switch 4 must be down to indicate a rescue operation to SOS. The computer must be inhibited or disabled while MXCORE is being loaded.

3.11.2 Output Requirements

MXCORE dumps the panel and all of core storage above the decimal location 3000 (the 3000 lower core locations are reserved for SOS control programs) onto B2 (see B2 format under subsection 3.11.4). SOS receives control and produces an output tape on A2 for printing.

3.11.3 Method

The method of operation for MXCORE is illustrated in Figure 3-6.

3.11.4 Usage

- a) Operator Procedures:
 - 1) Set sense switch 4 down
 - 2) Ready card reader
 - 3) Load MXCORE program
 - 4) Final program stop is at 178₈ (the standard SOS stop), with A2 containing the program dump for off-line printing.
- b) Format of B2 for SOS Rescue Dump—the execution of a CORE or PANEL macro causes the specified information to be written on B2 as

one or more physical records. Certain additional 1-word records are also written on B2 to serve as flags. The B2 for an SOS output dump has the following format:

Location of Record on B2	Length of Record	Description of Record		
First	One word	Flag signaling beginning of "snaps"		
Second	Nine words	PANEL macro output: AC, MQ, SI, XR1, XR2, XR4, ENK-keys, sense indicators, sense switches, AC overflow, divide check, and I/O check		
Third	Three words	CORE macro flag		
Fourth	Less than 256 words	Consecutive core storage locations, 255 or less		
Fifth	Three words	Record defines six or more consecutive identical storage words		
(The fourth record is repeated as many times as necessary to complete the limits of the CORE macro. The fifth record is repeated when six or more core storage locations have identical contents.)				
Last Record	One word	Flag signaling end of "snaps"		

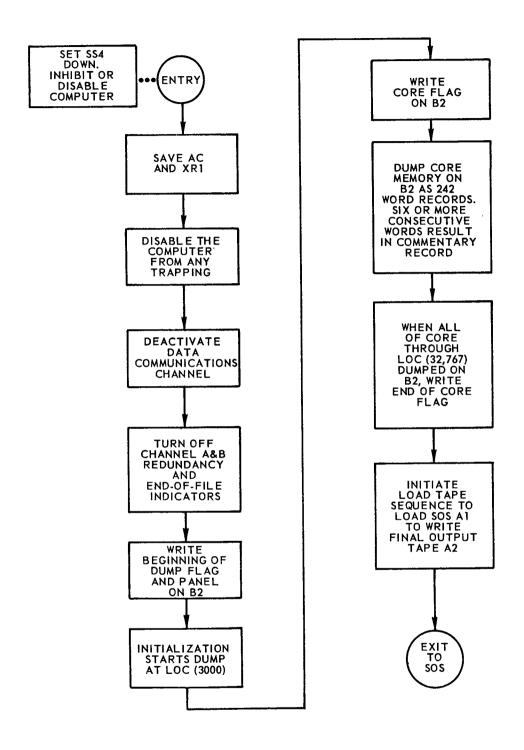


FIGURE 3-6. MXCORE PROGRAM FLOW CHART

3.12 SQUOZE TAPE MODIFICATION LIMITS PROGRAM (SUMARY)

SUMARY analyzes the squoze tape (or preface card of a squoze deck) of any SOS-assembled program and estimates the allowable number of modifications to that program for an SOS execution run. Also, if a squoze tape is used for input, SUMARY shows the "separation count"—the maximum such count for a successful recompilation is 32767.

The flow chart for SUMARY is shown in Figure 3-7.

3.12.1 Input Requirements

The fixed constant, currently 17004, for the particular SOS revision must follow the SUMARY program deck. The constant (in binary) must be in the address field of the 9-row left word on the data card. The remainder of the card is optional.

Either a row-binary preface card from the squoze deck of the program must be readied in the card reader after the data card, or the squoze tape of the program to be analyzed must be mounted on A5.

Several preface cards, squoze tapes, files on one squoze tape, or any combination thereof may be processed in succession by following the instructions printed on-line.

3.12.2 Output Requirements

SUMARY computes the number of core locations available for mod processing (called the mod packet count limit) and then, using a rule of thumb described under Method below, estimates the maximum number of cards the mod deck may contain. Additionally, if the input was a squoze tape, SUMARY shows the separation count.

All output from SUMARY is on-line. Thus, in the time it takes the operator to rewind and remove the tapes used for the compilation that created the squoze tape or deck (the symbolic listing, library, SOS, and intermediate tapes), SUMARY can be executed, requiring only the squoze tape or one squoze card.

3.12.3 Method

Rule of Thumb—approximate maximum number of allowable mod cards for Project Mercury will equal three-tenths of the mod packet count limit. Therefore, every three cards in the mod deck will require ten locations for processing. This rule of thumb has been accepted because

experience validates the following assumptions:

- 1) Ordinarily ALTER is used for the vast majority of modify and load pseudo-instructions. Some CHANGE cards are used.
- 2) Very few remarks cards and list control pseudo-instructions (SPACE, EJECT, DETAIL, etc.) are inserted for execution runs.
- 3) Essentially all inserted principal and generative pseudo-instructions (the latter category including DEC, OCT, BCI) will have location symbols.
- 4) The symbolic insertions cause essentially no additional "doubly-defined" symbol errors.
- 5) Every symbolic insertion makes, on the average, slightly more than one symbolic reference in its variable field.
- 6) The number and type of items deleted from the program by the mod deck has no effect on the size of the possible mod deck, except for the number of ALTER cards.
- 7) Few of the ALTER cards call for pure deletion. Most are either deletion with replacement or pure insertions.
- 8) The number of core locations required for processing mods is: 4 for each ALTER card, 2 for each inserted remarks card and inserted SOS pseudo-instruction of any kind (principal, generative, or list-control), 1 for each inserted location symbol, and 1 for each symbolic reference in the variable field of an inserted symbolic card.

Therefore, a typical 300-card mod deck for the Mercury Programming System might include:

Type of Cards	No. of Cards	No. of Locations Required
ALTER	20	80
ADIEN		
CHANGE	5	40
Machine and macro instructions	70	-
Remarks and list-control pseudo- instructions	5	10
Principal pseudo-instructions	100	200
Generative pseudo-instructions	100	200

Location symbols attached to instructions, macros, and pseudo-instructions	-	150
Symbolic references in the variable fields of inserted symbolic cards	<u>-</u>	320
	300	1000

In observing the relationships above, certain procedures for optimizing the mod deck become apparent. For example, it would be prudent to replace a BSS 1 with a PZE 0. It would be advantageous to gather all remarks cards for one routine in one place, perhaps at the start of the routine, and confine other comments to notes alongside the instructions rather than interspersing isolated remarks cards throughout the routine. It would pay to conserve the use of symbols and certain pseudo-instructions such as EJECT. But most apparent, it would help greatly to eliminate CHANGE and to reduce the number of ALTER cards. It would be better to say ALTER 3,6 and reinsert the two instructions formerly at alter numbers 4 and 5 than to say ALTER 3,3 and ALTER 6,6.

b) The separation count is the total number of machine instructions (including those generated by the generative pseudo-instructions, such as LBR and MACRO plus the number of principal pseudo-instructions). This quantity appears in the address of the last word of the second dictionary in a squoze deck. If the SUMARY input was a squoze tape (SSW#2 up), the tape is advanced to pick up this word and read it out. The maximum value of the separation count is 32767; if this limit is exceeded a recompilation of the program fails and SOS prints out the last symbolic card processed and the message COUNT TOO HIGH, COMPILATION STOPPED. Thus, for large programs such as CADFISS or MERCURY OPERATIONAL SYSTEMS, the difference between this number and the limit is important in planning additions or revisions of the system and changes to the scheme for multicompilations of the system.

3.12.4 Usage

One of the limitations imposed on the Mercury Programming System by SOS is the maximum size of the modification deck for an SOS Load and Go run. The length of time a compilation can be used before recompiling depends upon the number of modifications which can be made. To increase the life of a compilation, the multi-compilation scheme is provided. The size of the modification deck is dependent on:

a) The squoze tape size with respect to the number of symbols, pseudo-instructions, and other SOS quantities.

- b) The version of the SOS tape being used for the executive runs.
- c) The type of modifications being attempted.

The quantities from the squoze tape, which are used by SOS to allocate core storage for tables to process the mods, appear on the first card of the squoze deck, called the preface card. This card may be the first or second record of each file on the squoze tape, depending on whether a replica of the JOB card was written at the start of the squoze tape file by the particular version of SOS. Stacked squoze decks from successive CPL or PS runs are separate by an EOF mark on the squoze tape.

The number of modifications which may be processed in the available core storage space is also a function of the space required in core storage by the SOS programs. As previously stated, this is presently a fixed constant of 17004_{10} . The constant for any given SOS tape is computed as follows:

(All numbers are decimal except location numbers which are in parentheses and refer to the contents of the address or decrement of those locations when tested by the M3 section of SOS)

D = number of dictionary entries

I = number of introduction words

F = number of footnote words

NL = 2's complement of next location for storing squoze text

M = the algebraic sum of all quantities considered from the mod deck called the "mod packet count limit."

Using the Mercury SOS tape and an ordinary mod deck (which has no ERASE, ASSIGN, or SYMBOL pseudo-ops), an error message, MODIFICATIONS EXCEED LIMITS will be printed if:

$$c(77722)_{a} + c(77722)_{d} \ge c(77730)_{a} - \left[74 + c(77637)_{d} + c(77730)_{d} + \frac{3}{2}c(77731)_{d} + c(77734)_{d} + c(77735)_{d} + c(77736)_{d} + 2c(77740)_{d}\right].$$

where c(77722)_a = 2D + F + 3930, therefore fixed by the compilation. The constant 3930 is fixed for a given SOS tape; in the unmodified IB version from New York it is 5030.

 $c(77722)_d = 1$ for each symbol in the variable field of mods.

 $c(77730)_a = -10000 (=22768)$, a constant for a given SOS tape.

 $c(77637)_d = I$ from the compilation.

c(77730)_d = NL: maximum 1760, minimum 1530, not under programmer control.

 $c(77731)_d = 2$ for each ALTER and CHANGE in the mod packet. Multiplied by 3/2 in the above inequality.

c(7734)_d = 2 for each EQU, SYN, or BOOL in the mod deck.

c(77735)_d = I inserted by the mod packet.

 $c(77736)_d$ = F added by the mod packet.

c(77740)_d = 2 for each CHANGE and for each principal pseudo-instruction without a location symbol; 1 for each ALTER, 1 for each location symbol; and 1 for each principal pseudo-instruction.

Multiplied by 2 in the above inequality.

74 = constant for a given SOS tape.

Immediately after the compilation is performed, since D, F, I, and the constants 3930, 22768, and 74 are known, the original inequality reduces to:

$$(2D + F + 3930) \ge 22768 - (I + 74 + M + NL).$$

That is, the failure will occur if M reaches (18764 - 2D - F - I - NL). Since the maximum value of NL is 1760 and the actual instantaneous value is not predictable, the safe limit is reduced to 17004 - (2D + F + I).

An SOS modification deck consists of one or more Modify and Load pseudo-instructions (ALTER, CHANGE, ERASE, ASSIGN, SYMBOL), plus any desired insertions to the program in the form of symbolic cards which may include machine instructions, pseudo-instructions, macro instructions, and symbolic locations and comments.

Certain combinations of these require more storage area for processing than others. By judicious choice and arrangement of the pseudo-instructions and symbolic insertions, the mod deck can be optimized, greatly increasing the allowable number of modifications. Conversely, very small mod decks can be produced which quickly exhaust the available storage.

a) Operator Procedures:

- 1) Ready the SUMARY binary program deck with data card behind it in the on-line card reader.
- 2) Either ready a squoze tape on A5 (it will be there already if created during an SOS CPL or PS job with SSW#6 up) or ready the preface card of a row-binary squoze deck in the card reader behind the data card.

- 3) Press CLEAR and LOAD CARDS.
- 4) An HPR should occur almost immediately after an on-line printer message (requesting either SSW 2 be depressed or the tape number of the squoze tape be entered in the console keys in BCD, right-justified) is printed. Tape number 285 would then be entered as 000000021005. After setting the sense switch or keys, press START.
- 5) After the SUMARY analysis has been printed, a halt occurs to permit the operator to select one of these options:
 - (a) SSW #2 down if a preface card in the card reader is to be analyzed.
 - (b) SSW #1 up if finished or if the operator wants to analyze the first job on a squoze tape.
 - (c) SSW #1 down to analyze the next job on the same squoze tape as the one just processed.

After selecting the option, press START. If (a) were chosen, repeat procedure from step (4).

- 6) A final halt occurs:
 - (a) If finished, press CLEAR and retrieve the SUMARY deck from the card reader.
 - (b) To analyze the first job on another A5 tape, ready that tape and press START. Repeat from step (4).
- b) Error Conditions—an HTR occurs after ten unsuccessful attempts to read A5. Check the density setting and press START to accept the next attempt and continue.
- c) Interpretation of Output—while SOS quantities listed in the SUMARY analysis are exact, the estimate of the number of mods possible is an approximation and should be used only as a guide to plan the time for recompiling.

Moreover, it should be emphasized that the SUMARY program was created to aid in avoiding only two known SOS limits—1) having the job rejected by SOS because modifications exceed limits; 2) having a compilation fail because of a COUNT TOO HIGH.

There are several other limits and restrictions which apply; among these are: (1) the fixed number of dictionary entries permitted (8000),

(2) the fixed number of footnotes (principal pseudo-instructions) which may be inserted (1000), (3) the number (20) and size (about 120 locations) of programmer macros which may be redefined or defined at load-and-go-time, and (4) the prohibitions against altering in or out any HEAD cards, or altering out any remarks of list-control pseudo-instructions, or altering in any LBR cards.

d) Subroutines Used:

- 1) SUBR-performs a binary-to-BCD conversion for integers
- 2) SUPRES-suppresses leading zeros from a BCD word
- 3) CAP-converts BCD to Hollerith card image and reads out one line on-line, maximum of 72 characters
- 4) SKPUP—given a word count W = 23N + r, reads from A5 N records if r = 0, N + 1 records if $r \neq 0$.

e) Notes:

The following notes pertain to the SUMARY program flow chart, Figure 3-7.

Block 1—initialization consists of reading the data card with the SOS constant, storing the constant, and ejecting a page on the printer.

Block 2-self explanatory.

Block 3—CAP is entered three times to print a three-line request of the operator to enter the number of the squoze tape in the keys or, if input is from the card reader, to depress sense switch #2. The printer then ejects a page and the computer halts until the operator presses START.

Blocks 4-6-self explanatory.

Block 7—one record is read from the squoze tape.

Block 8-self explanatory.

Block 9-the redundancy counter is reset to the maximum value, ten.

Block 10—some squoze tapes start each file with a replica of the JOB card ahead of the preface card. This must be bypassed by SUMARY. The 9L word of the preface card (and any squoze card) is minus; a plus word is assumed to be a JOB card.

Blocks 11-13-self explanatory.

<u>Block 14</u>—CAP is entered twice to print a 2-line message informing the operator that ten successive redundancies occurred while trying to read one record from A5.

Block 15—if START is pressed, an eleventh attempt is made and accepted, and processing continues.

Block 16—the 12 BCD characters from columns 16-27 of the original JOB card are obtained from the preface card and placed in a line image which CAP then processes.

Block 17—the compilation date is obtained from the preface card and CAP is entered to print four lines—the date, a blank line, a note that all numbers are decimal rather than octal, and another blank line.

Block 18—each of the following eight SOS quantities are obtained from the preface card and read out using the subroutines SUBR, SUPRES, and CAP:

- D—number of dictionary entries (symbols and principal pseudo-instructions).
- I—number of introduction words (generative pseudo-instructions and first of each sequence of remarks cards and list-control pseudo-instructions).
- F—number of footnote words (principal pseudo-instructions except HEAD).

Number of words of noncommentary text (zero with squoze tapes produced by the current SOS).

Number of words of text with commentary.

Number of programmer macros.

Number of HEAD cards.

Number of alter numbers.

Block 19—the quantity (2D + F + I) is computed and read out, using the subroutines SUBR, SUPRES, and CAP.

Block 20—the SOS constant less the quantity (2D + F + I) is called the mod packet count limit and is the number of core locations available for the mod processing. It is computed and printed using the subroutines

SUBR, SUPRES, and CAP. A blank line is then printed. The maximum number of mod cards possible for an SOS execution run is equated to three tenths times the mod packet count limit computed above, using the rule of thumb described earlier. This number is computed and printed, using the subroutines SUBR, SUPRES, and CAP.

Block 21-self-explanatory.

Block 22—when input is a squoze tape (rather than cards) the additional quantity "separation count" described above is computed and printed out, using the subroutines SKPUP, SUBR, SUPRES, and CAP.

Block 23—two blank lines, five explanatory comments, and three more blank lines are printed, using CAP for each line. If input was tape, a sixth comment concerning the separation count limitation on the compiler is printed before the three final blank lines.

Blocks 23-25-self-explanatory.

Block 26—a page is ejected on the on-line printer and the program instructs the operator with a 2-line message (using CAP twice) as follows:

PRESS START TO ANALYZE NEXT PREFACE CARD.

IF DONE, PRESS CLEAR. RETRIEVE THE SUMARY DECK FROM THE CARD READER.

The printer skips three blank lines and the computer comes to a final halt. The operator should either press START, returning control to the initialization section of SUMARY, or CLEAR, and get off the machine.

Blocks 27-28—the number of the squoze tape, which the operator should have entered in the console keys, is printed in a final comment after which a page is ejected on the on-line printer.

Block 29—the program notifies the operator of the following option and then halts:

Depress SS 1 to analyze the next job on the same squoze tape,

Leave SS 1 up to analyze the first job on another squoze tape, to analyze a preface card, or if the run is complete.

Blocks 30-31-self-explanatory.

<u>Block 32</u>—the program notifies the operator of the following options and then comes to a final halt:

If the run is completed, press CLEAR and retrieve the SUMARY deck from the card reader (as well as the output from the on-line printer).

If the first job on another squoze tape is to be analyzed, that tape should be readied and START pressed, returning control to the initialization section of SUMARY. At this point, card input may be introduced by readying row-binary preface card(s) in the card reader and pressing START. Later sense switch #2 will have to be depressed to designate card input.

Block 33-self-explanatory.

<u>Blocks 34-38</u>—either one or two EOF marks may separate stacked jobs on the squoze tape. This routine positions the tape for the next job regardless of which condition exists.

Block 39-self-explanatory.

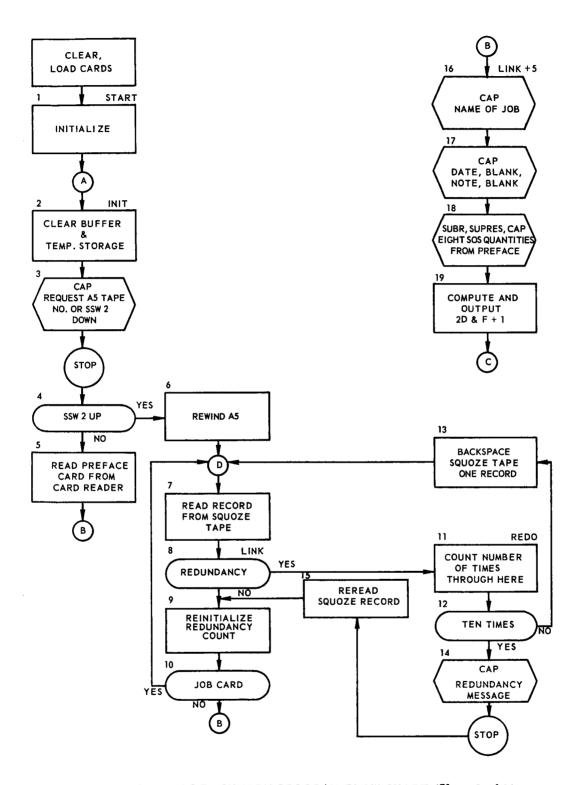


FIGURE 3-7. SUMARY PROGRAM FLOW CHART (Sheet 1 of 2)

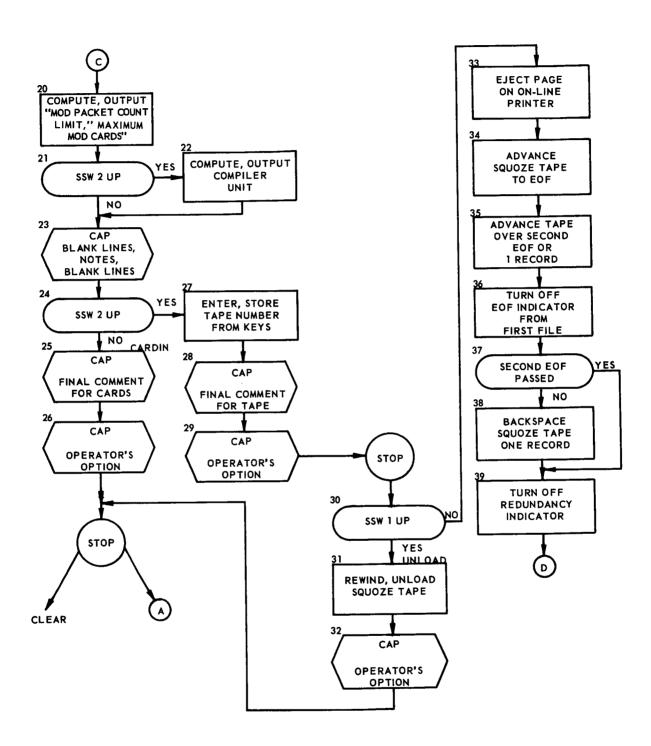


FIGURE 3.7. SUMARY PROGRAM FLOW CHART (Sheet 2 of 2)

3.13 PAPER TAPE INPUT PREPARATION (PAPTAP)

PAPTAP is the common name of two independent programs, PAPTAP-A and PAPTAP-B which are used in the preparation of paper tape inputs, in teletype coding, for the Mercury tracking program. These paper tapes are used to simulate receipt of radar data. They are fed to the tracking program from local tape readers (ASR's) or they may be shipped or transmitted to radar sites for long-distance retransmission to the computer.

3.13.1 Method

The source data for PAPTAP is a magnetic tape prepared by the Mercury simulation programs for use by SIC. This input tape contains the radar data needed. In the first of two runs (using PAPTAP-A), the radar is properly formatted and is punched on paper tape. In the second run (using PAPTAP-B), the SIC input tape is compared with the punched paper tape. When a discrepancy is found, the program stops and displays the error in the A-register.

The only comments which may be needed for an understanding of the flow charts (Figures 3-8 and 3-9) are these: A complete radar transmission always begins with J, J, LRTS, CR, LF, LTRS; and always ends with BLANK, FIGS, H, LTRS. The two key points for the programs are the initial J and the final H. Both programs search for the initial J and, having found it, output it (or compare it) and succeeding characters until the H is found. Having found the H, they assume that the next character is LTRS, and therefore immediately begin searching for the next J.

Since the programs depend upon finding the initial J and final H, trouble may occur when radar data with simulated teletype errors is used. That is, it may well be that the 5-bit coding for H will appear (as a simulated error) before the actual end of transmission, or that the final H will not occur at all. In short, unless greater redundancy is built into the program (e.g., searching for five out of six of the characters J, J, LTRS, CR, LF, LTRS for the beginning of a transmission and a corresponding scheme for the end of transmission), it is possible to produce a tape with incorrect data.

3.13.2 Magnetic Tape Formats

The first record on tape is a label, and it is skipped over by the program. Each succeeding tape record contains 22 logical records. The appearance of each logical record in the memory of the IBM 7094 (which was used to produce the magnetic tape) and in the memory of the CDC-160 is illustrated below.

IBM 7094:

		36 bits		-
Word 1	7	Time of ar	rival	
2	No. of wo	rds and su	bchannel No.	
3	T	ime for int	terrupt	
4	31	zeros	xxxxx	
5	31	zeros	xxxxx	
6	31	zeros	xxxxx	6 teletype
7	31	zeros	xxxxx	characters
8	31	zeros	xxxxx	
9	31	zeros	xxxxx] /

End of logical record

CDC-160:

	12 bits	
Word 1-6	Time of arrival	Each 7094 6-bit
7-12	No. of words and subchannel No.	byte is preceded by 6 zeros.
13-18	Time for interrupt) by o zeros.
19-23	All zeros	
24	0000000XXXXX	First TTY character
25-29	All zeros	
30	0000000XXXXX	Second TTY character
31-35	All zeros	
36	0000000XXXXX	Third TTY character
37-41	All zeros	

42	0000000XXXXX	Fourth TTY character
43-47	All zeros	
48	0000000XXXXX	Fifth TTY character
49-53	All zeros	
54	0000000XXXXX	Sixth TTY character

End of logical record

3.13.3 Paper Tape Formats

The paper tape produced by PAPTAP has the usual teletype radar format illustrated below. This format differs from the sequence of teletype characters on magnetic tape in one respect only: teletype characters appearing on magnetic tape between the end of one transmission and the beginning of the next are not punched on the paper tape.

The output format consists of three sections: preamble, variable number of radar reports, and end-of-transmission sequence. These sections immediately follow one another.

- a) The preamble consists of the six characters J, J, LTRS, CR, LF, LTRS.
- b) Each radar report consists of:

Characters 1-3	CR, LF, FIGS
4	Kind of data
5-6	Station identification
7	Radar type
8	Data validity
9-14	Time in hours, minutes and seconds
15-20	Azimuth
21-26	Elevation
27-33	Range
34	OBLIQUE STROKE

c) The end of transmission sequence consists of the four characters BLANK, FIGS, H, LTRS.

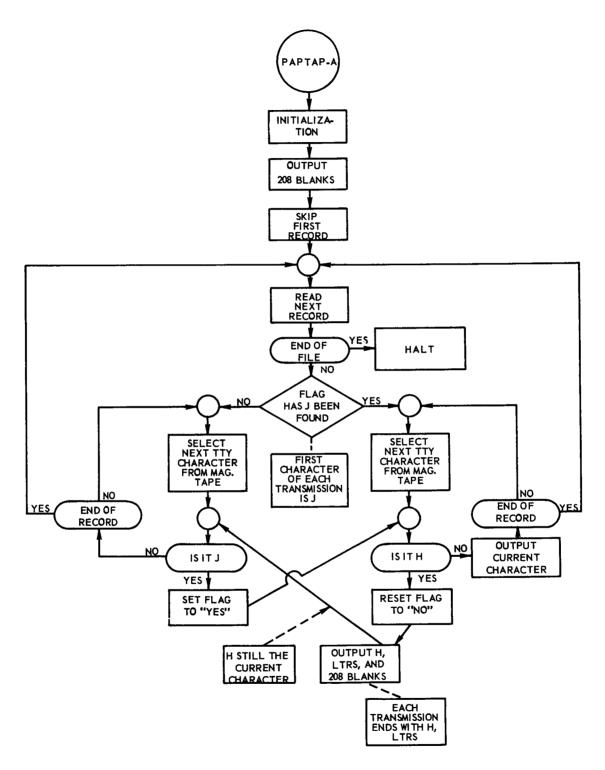


FIGURE 3-8. PAPTAP. A PROGRAM FLOW CHART

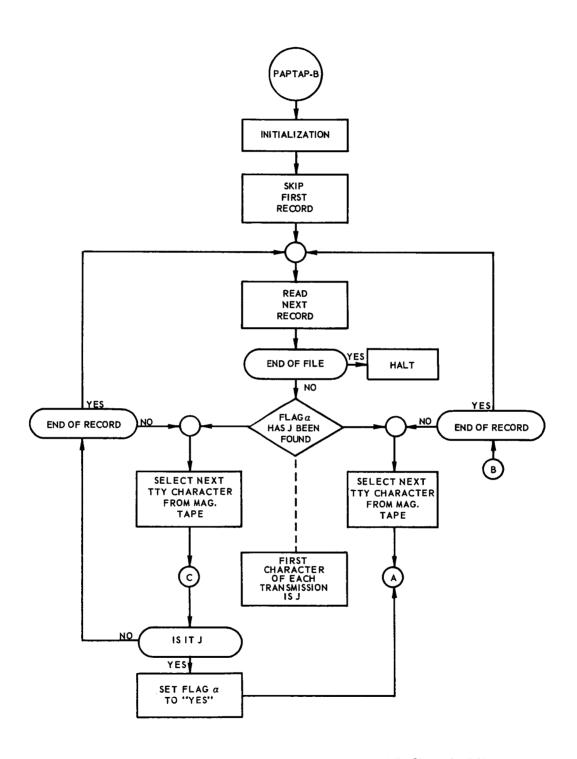


FIGURE 3-9. PAPTAP.B PROGRAM FLOW CHART (Sheet 1 of 2)

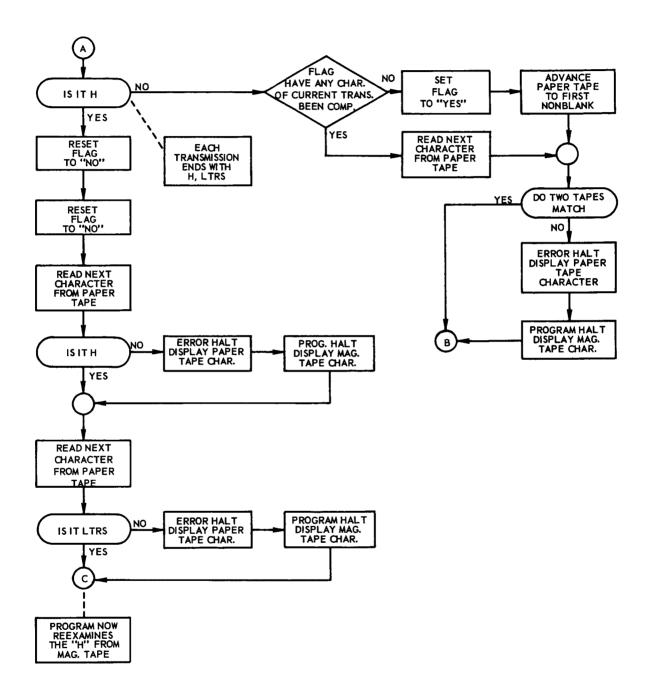


FIGURE 3-9. PAPTAP.B PROGRAM FLOW CHART (Sheet 2 of 2)

3.14 LOW-SPEED OUTPUT PRINTER PROGRAM (MXTHLG)

MXTHLG examines the MXPRLG output tape for low-speed TTY data received from Mercury radar stations and unpacks, converts, and formats it for off-line printing.

The flow chart for MXTHLG is shown in Figure 3-10.

3.14.1 Input Requirements

Input to MXTHLG is the output tape from the MXPRLG program for a Mercury mission, simulated or unsimulated, and a 4-card deck prepared for the online card reader. The cards contain internal station numbers, a density mutilation coefficient constant, and the ID for the data (mission).

3.14.2 Output Requirements

The output tape A3 will contain in decimal form all low-speed input data received at Goddard during a mission. When printed, each line of data from the output tape contains a radar message listing: kind of data, internal station number, valid, type of radar, time of message (hours, minutes and seconds), range, azimuth, and elevation. For example:

KIND OF DATA	INTERNAL STA. NO.	TYPE OF RADAR	VALID	HR.	TIME MIN.	SEC.	AZIMUTH (degrees)
X	XX	X	X	XX	XX	XX	XXX.XXX
		ELEVATI			RAN (yar		
		XXX.XX	X	XX	XXXXX	xx.xx	

3.14.3 Method

MXTHLG reads all data from the MXPRLG output tape, separating the low-speed data from other messages, and edits and writes this on an auxiliary tape to produce a low-speed TTY tape. The program reads data from the TTY tape, repacking and converting it to the correct format, and then writes the reformatted data on the output tape.

3.14.4 Usage

Operator's Procedure:

- a) Ready A4 with an output tape from MXPRLG.
- b) Ready A3, A5 with blank tapes.
- c) Ready A9 with FORTRAN written station characteristic tape.
- d) Ready C1 program tape (postflight).
- e) Ready cards in on-line card reader.
- f) Sense Switches 1 and 7 down.*
- g) Press CLEAR and LOAD CARDS buttons.
- h) Print A3 under program control.

^{*}Other Sense Switches offer various options (see flow chart).

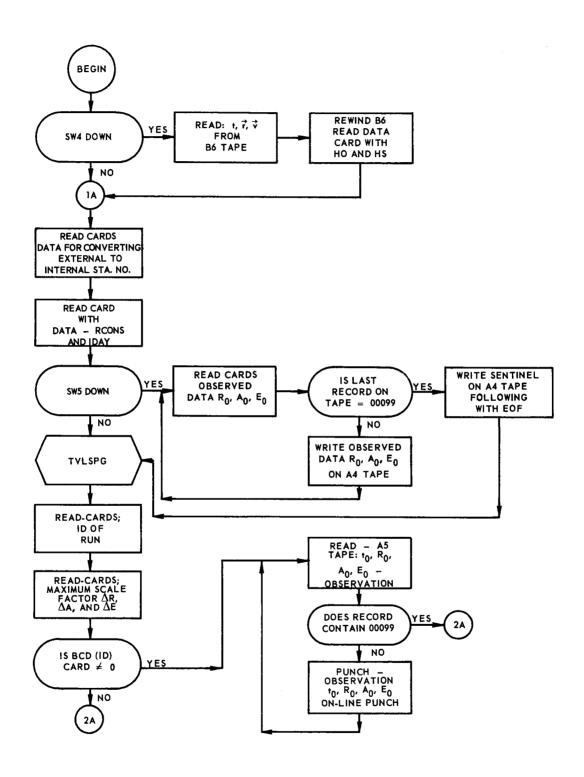


FIGURE 3-10. MXTHLG PROGRAM FLOW CHART (Sheet 1 of 10)

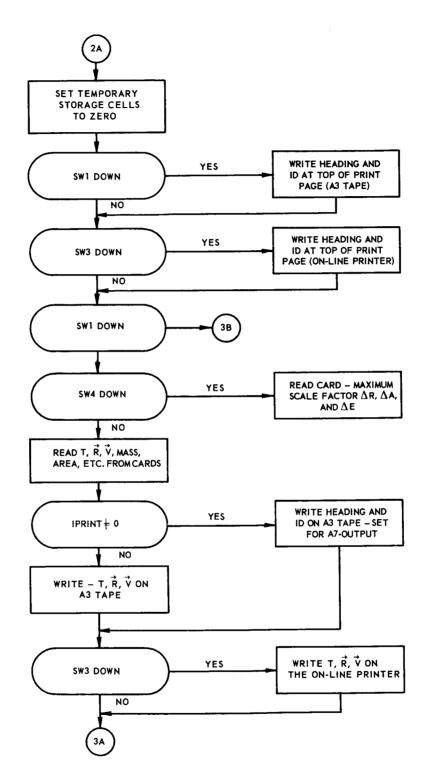


FIGURE 3-10. MXTHLG PROGRAM FLOW CHART (Sheet 2 of 10)

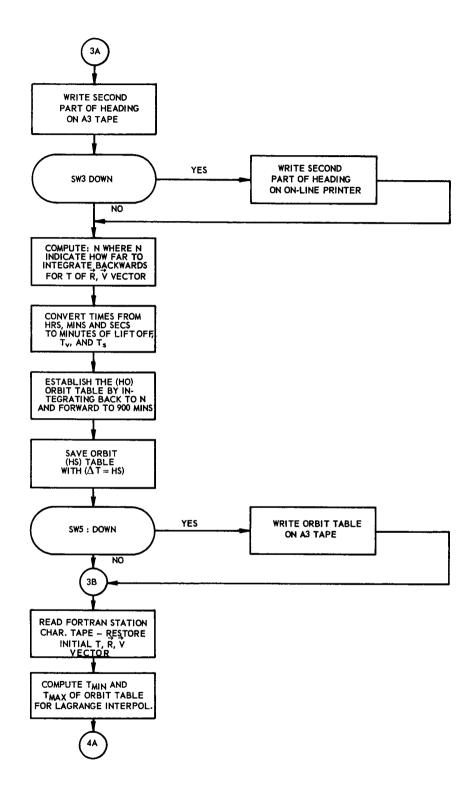


FIGURE 3-10. MXTHLG PROGRAM FLOW CHART (Sheet 3 of 10)

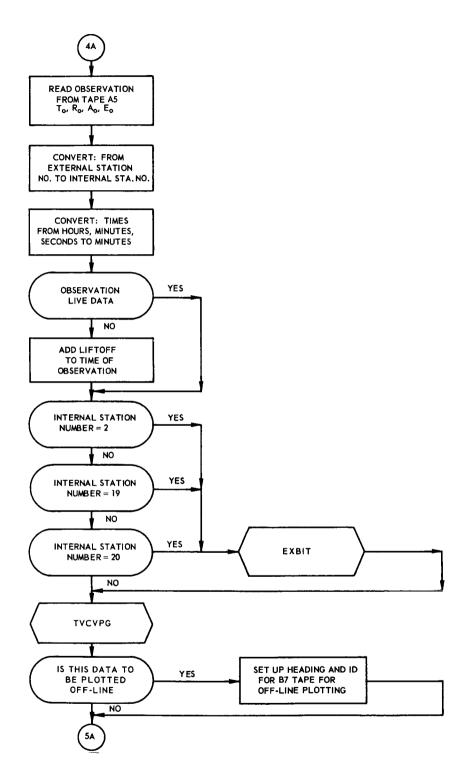


FIGURE 3-10. MXTHLG PROGRAM FLOW CHART (Sheet 4 of 10)

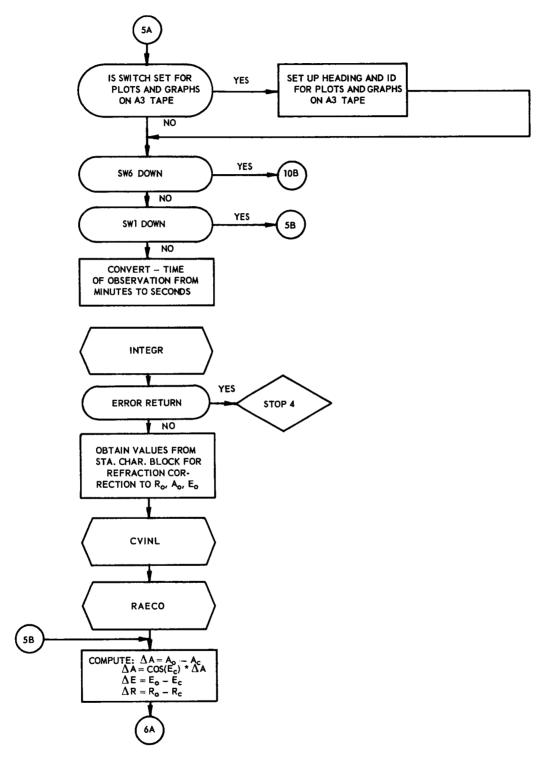


FIGURE 3-10. MXTHLG PROGRAM FLOW CHART (Sheet 5 of 10)

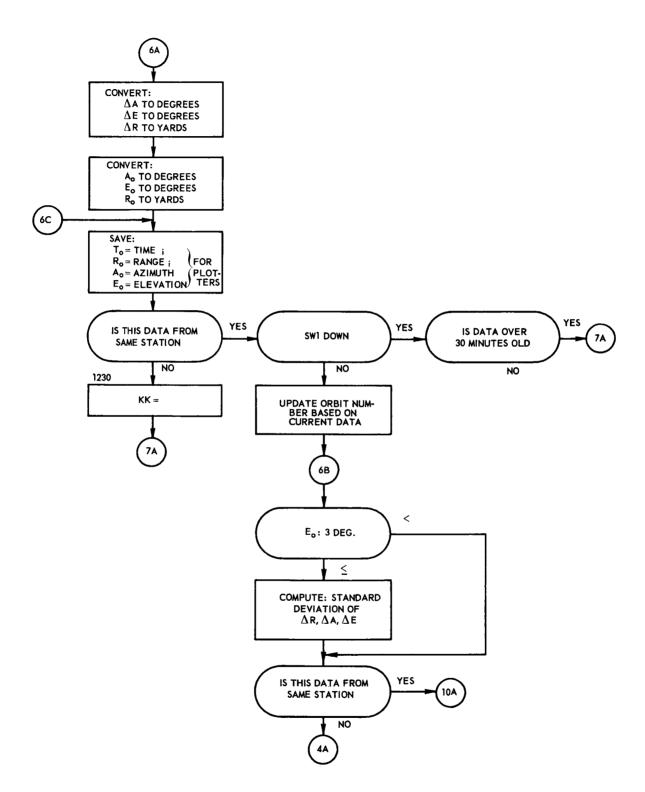


FIGURE 3-10. MXTHLG PROGRAM FLOW CHART (Sheet 6 of 10)

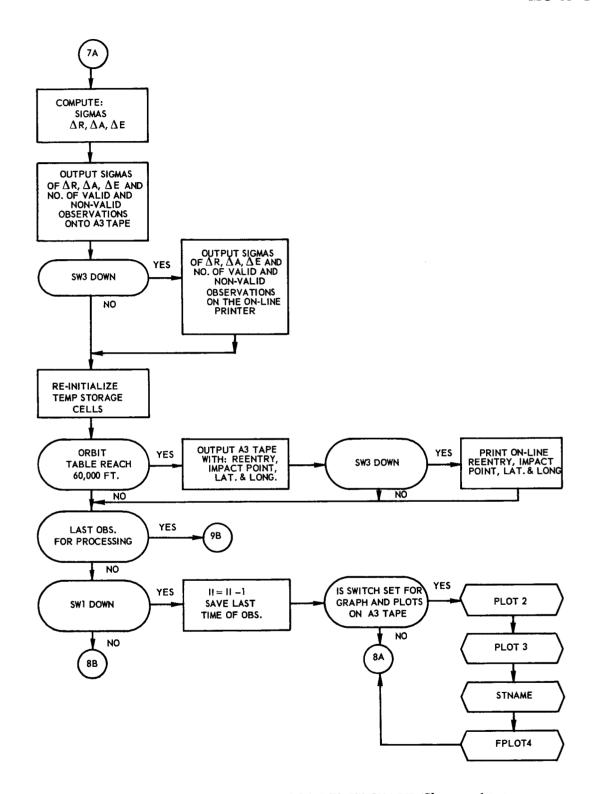


FIGURE 3-10. MXTHLG PROGRAM FLOW CHART (Sheet 7 of 10)

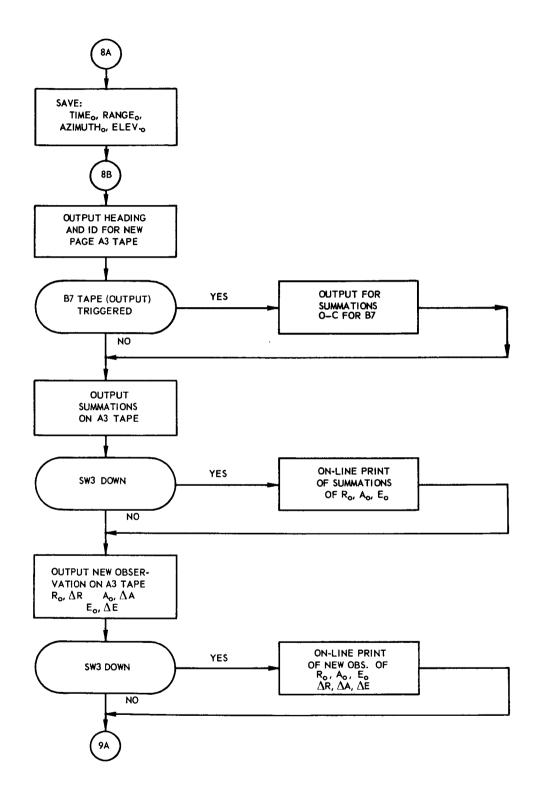


FIGURE 3-10. MXTHLG PROGRAM FLOW CHART (Sheet 8 of 10)

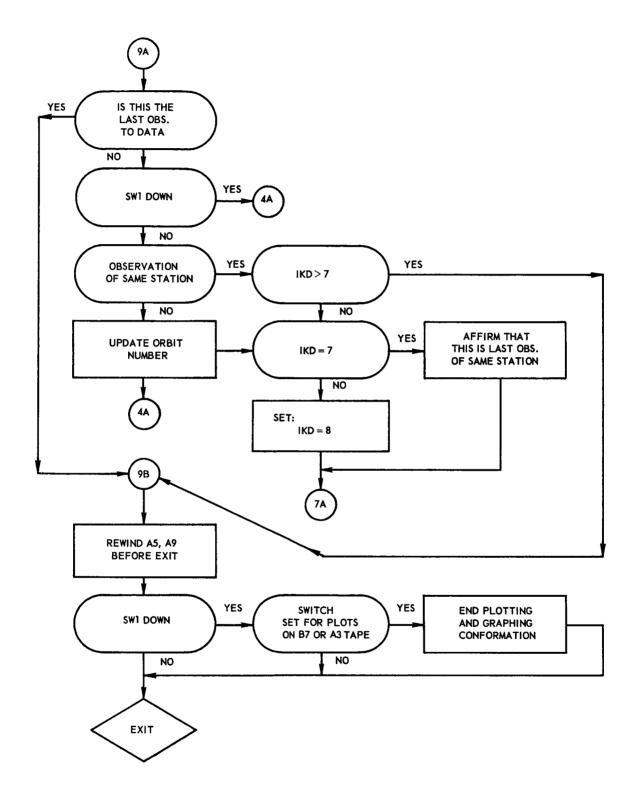


FIGURE 3-10. MXTHLG PROGRAM FLOW CHART (Sheet 9 of 10)

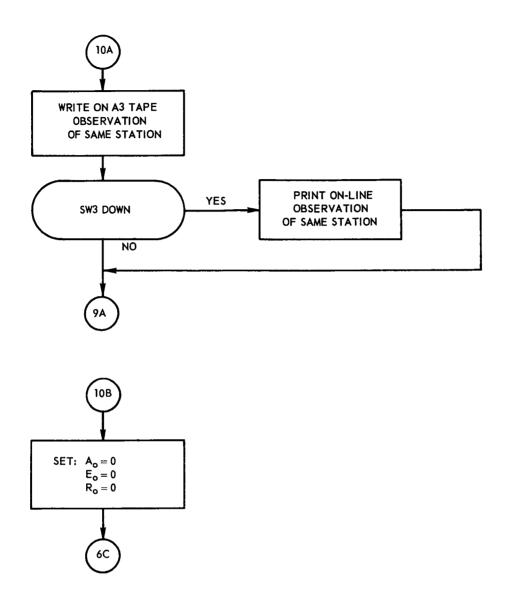


FIGURE 3-10. MXTHLG PROGRAM FLOW CHART (Sheet 10 of 10)

3.15 LOG TAPE HIGH-SPEED INPUT PROGRAM (HSIN7)

HSIN7 examines the log tape for high-speed IP 7094, B-GE, or Bermuda radar input messages.

The flow chart for HSIN7 is shown in Figure 3-11.

3.15.1 Input Requirements

The only input to HSIN7 is the log tape from a Mercury mission—simulated or unsimulated.

A log tape is composed of one file which contains up to 9286 logging buffers; each buffer is one record and is composed of ten 17-word logging blocks. Each logging block begins with a 5-word heading; the remaining 12 words contain either logged data or a nondata mask. The heading contains a data identification by the DCC subchannel, an indication of whether the data was transmitted, a data word count, a logging block serial count, the mission phase, and two time tags. One time tag is associated with the data itself and the other specifies the time the data was logged.

3.15.2 Output Requirements

The output tape contains the selected high-speed messages printed in tabular form. Figure 3-12 shows the output of HSIN7. The heading on each output page indicates IP 7094, B-GE, or Bermuda radar input followed by a heading above each column. An explanation of the column headings is given below.

- a) Logging Time—the time, in seconds, shown by the internal clock at time of trap. If the internal clock were synchronized with Greenwich Mean Time (GMT), this column would show GMT. In the example, the message was logged at 44387.423 (or 12 hours), 19 minutes, and 44.4 seconds.
- b) Message Time—the time associated with each of the incoming position and velocity vectors. It is incorporated within the input message. In the example in the table, the time of the position and velocity vectors is 275.000 seconds after liftoff.
- c) X, Y, and Z—indicate the components of the position vector using the appropriate units of the input source.
- d) X, Y, and Z—indicate the components of the velocity vector using the appropriate units of the input source.

- e) Discrete Signals—in the B-GE section, the discrete signal column gives an octal representation of eight binary bits. Each of the binary bits is contained somewhere within the input message. The first bit indicates that liftoff has occurred, the next four bits are data quality flags, the next two bits indicate Booster Engine Cutoff (BECO) and Sustainer Engine Cutoff (SECO, respectively, and the last bit gives the B-GE, GO-NO-GO recommendation.
- f) Checksum—if the checksum in the input messages is identical to the computed checksum, this column contains a zero; otherwise, it contains a one.
- g) Parity—the correct format for input messages is such that the parity is always odd. An odd parity is indicated by a one in this column. A zero indicates incorrect parity.

Table 3-1 gives the quantity associated with bits 1 through 72 of the telemetry format shown in Figure 3-12.

3.15.3 Method

(Not applicable.)

3.15.4 Operator's Procedures

- a) Ready the standard SOS system tapes.
- b) Ready the log tape on B6.
- c) Ready a blank on A3 for output.
- d) Ready the HSIN7 deck
- e) Press CLEAR and LOAD CARDS.
- f) Program will print operating instructions on line.

Set Keys = 1 for B/GE

Set Keys = 2 for IP 7094

Set Keys = 3 for BDA radar in radians.

Set Keys = 4 for BDA radar in degrees.

g) If a special binary output of B/GE or IP is desired set appropriate entry key, place a blank tape on A4 and depress sense switch 1.

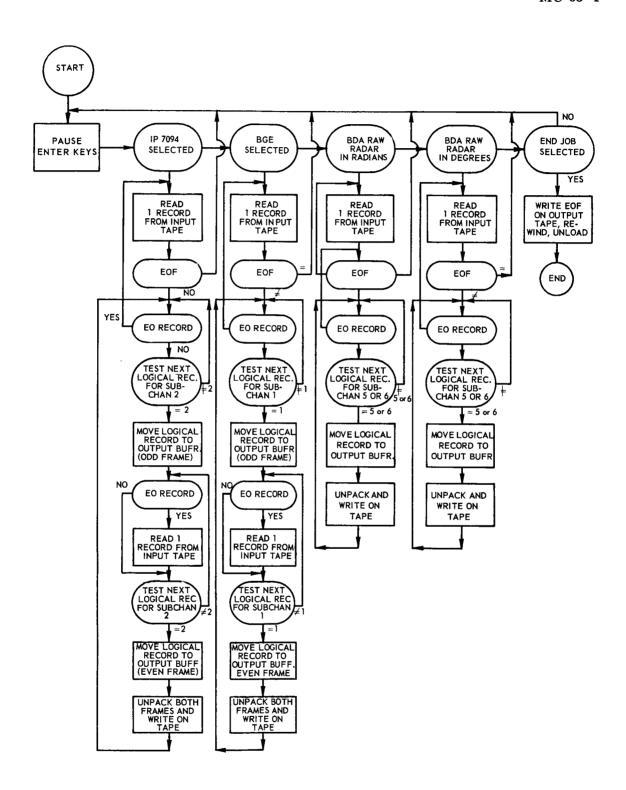


FIGURE 3-11. HSIN7 PROGRAM FLOW CHART

1 P 7 0 9 4

ZA (E	
AZUZ SP I	00
SORC BIT	00
77	5 5
C P 666666667 1234567890	7 0 1 7 0 1 0000000011 0000000011
Z DOT C P 555555556 66666 1234567890 12345	0.154021 0.154021 1000010111
X DOT Y DOT 3333334 44444445 4567890 1234567890	0.2546107 0.2546107 0000000011 0000000011
X DOT 333333334 1234567890	0.6594161 0.6594161 0000000000 0000000000
222222233 1234567890	0.5095008 0.5095008 0000000001
X Y Z X DOT 7 DOT C P 1 111111111 222222223 333333334 4444444445 555555556 666666667 77 SORC AZUZA 1234567890 1234567890 1234567890 1234567890 1234567890 12 BIT SPIKE	14387.423 275.000 0.2331927 -0.8573266 0.5095008 0.6594161 0.2546107 0.1540217 0 1 14387.423 275.000 0.2331927 -0.8573266 0.5095008 0.6594161 0.2546107 0.1540217 0 1 0010000000 010000000 0000000001 00000000
X 1 234567890	0.2331927 0.2331927 010000000
M TIME	275.000 275.000 0
L TIME TELEMETRY FORMAT	44387.423 44387.423

B / G E

X DOT Y DOT Z DOT DSC C P	1 1111111112 222222223 33333334 444444445 555555556 666666667 77	234567890 1234567890 1234567890 1234567890 1234567890 1234567890 1234567890 12	44455,141 341.964 0.9018390 0.1641586 0.5431135-1.5318134 0.8371127-0.9550688 017 0 1 44455,141 341.964 0.9018390 0.1641586 0.5431135-1.5318134 0.8371127-0.9550688 017 0 1 0010000000 010000000 0000000000	0010000000 0100000000 0000000001 0000000
×	22222233	1234567890 1	.5431135-1.5 .5431135-1.5 0000000001	0000000001
>	11111111112	1234567890	0.1641586 0 0.1641586 0 0100000000	0100000000
×	-	234567890	0.9018390	010000000
L TIME M TIME		_	341.964 341.964 0	0
L TIME	TELEMETRY	FORMAT	44455.141 44455.141	

RADIANS **z** RADAR RAW BERMUDA

LOG TIME SUBCH MSG	SUBC	H MSG TIME	VOT	TIME VOT V RANGE V AZIMUTH V ELEV FOT F RANGE	V AZIMUTH	V ELEV	FOT F	RANGE	F AZIMUTH F ELEV BE/CE GE/CE BE/GE	F ELEV	BE/CE	GE/CE	3E/GE
4486.308	ស	4486.308 5 12*21*26.1 1	-	1206480.	4.368394	1206480, 4,368394 0,431432 1 1207643, 4,369161 0,429467 0 0 0	-	1207643.	4,369161	0.429467	0	0	0
4486.316	•	5 12*21*26.1 1		1206480. 4.368394 0.431432 1 1207643. 4.369161 0.429467 0 0	4.368394	0.431432	-	1207643.	4.369161	0.429467	0	0	0
			8	BERMUDA RAW RADAR IN DEGREES	RAW	RADAR	_	N DEG	REES				

:/GE	0	0
BE/CE GE/CE BE/GE	0	0
CE GE		
BE/	0	0
F ELEV	11.2527 0	11.2527 0
F AZIMUTH F ELEV	255.6079	255.6079
FOT F RANGE	2182546.	2182546.
FOT	-	-
V ELEV	11,3654	11.3654
V AZIMUTH V ELEV	255.5475 11.3654	255,5475
VOT V RANGE	2181346.	2181346.
	-	-
LOG TIME SUBCH MSG TIME	5 12*20*42.9	12*20*42.9
SUBC	ស	9
IME.	108	116
L06 T	44443.108	4443.116

FIGURE 3-12. EXAMPLES OF LOGGING MESSAGE FORMATS FOR HSIN7

Table 3-1. TELEMETRY FORMAT, IP 7094 AND B-GE

Bit No.	Quantity	Comment
1	Selected source (1)	Note 1
2	Selected source (2)	Note 1
3	Selected source (3)	Note 1
4	Start BDA solution	1 = START
5	1 of 3 Retrogrades fired	1 = Has occurred
6	2 of 3 Retrogrades fired	1 = Has occurred
7	3 of 3 Retrogrades fired	1 = Has occurred
8-9	Spares	
10	Source selected (1)	Note 1
11	Source selected (2)	Note 1
12	Source selected (3)	Note 1
13	Start BDA solution	1 = START
14	1 of 3 Retrogrades fired	1 = Has occurred
15	2 of 3 Retrogrades fired	1 = Has occurred
16	3 of 3 Retrogrades fired	1 = Has occurred
17-25	Spares	
26	3 of 3 Retrogrades fired	1 = Has occurred
27	2 of 3 Retrogrades fired	1 = Has occurred
28	1 of 3 Retrogrades fired	1 = Has occurred
29	Start BDA solution	1 = START
30	Source selected (3)	Note 1
31	Source selected (2)	Note 1
32	Source selected (1)	Note 1
33-42	Spares	
43	1 of 3 Posigrades fired	1 = Has occurred, Note 2
44	2 of 3 Posigrades fired	1 = Has occurred, Note 2
45	3 of 3 Posigrades fired	1 = Has occurred, Note 2
46	1 of 3 Retrogrades fired	1 = Has occurred, Note 2
47	2 of 3 Retrogrades fired	1 = Has occurred, Note 2
48	3 of 3 Retrogrades fired Liftoff	1 = Has occurred, Note 2 1 = Has occurred
49 50	Escape tower released	_
51	Tower escape rockets fired	1 = Has occurred 1 = Has occurred
52	Spacecraft separation	1 = Has occurred
53	Abort sequence initiated	1 = Has occurred
54	Abort phase has started	1 = Has occurred
55	Orbit phase has started	1 = Has occurred
56	Selected source (3)	Note 3
57	Sustain engine cutoff	1 = Has occurred
58	Liftoff	1 = Has occurred
59	Escape tower released	1 = Has occurred
60	Tower escape rockets fired	1 = Has occurred

Table 3-1 (continued). TELEMETRY FORMAT, IP 7094 AND B-GE

Bit No.	Quantity	Comment
61	Spacecraft separation	1 = Has occurred
62	Abort sequence initiated	1 = Has occurred
63	Abort phase has started	1 = Has occurred
64	Orbit phase has started	1 = Has occurred
65	Orbit phase has started	1 = Has occurred
66	Abort phase has started	1 = Has occurred
67	Abort sequence initiated	1 = Has occurred
68	Spacecraft separation	1 = Has occurred
69	Tower escape rockets fired	1 = Has occurred
70	Escape tower released	1 = Has occurred
71	Liftoff	
72	Parity for previous 71 bits	1 = Even number of 1's 0 = Odd number of 1's

Notes:

1. Source selection is indicated by the following bit configurations:

	(Bits-1, 10, 32) Selected Source (1)	(Bits-2, 11, 31) Selected Source (2)	(Bits-3, 12, 30, 56) Selected Source (3)
B-GE	0	0	1
IP 7094	1	0	0
BDA	0	1	0

- 2. Retrogrades TLM bits 46, 47 and 48 will remain until GSFC has made the program changeover. These bits will then be disconnected and used as spares.
- 3. Selected source (3) will be wired to bit 56. This arrangement has a twofold purpose: (1) enable the program to use the new TLM format without implementing a program change, and (2) if they inadvertently selected Bermuda without Short Arc incorporation they will automatically select IP 7094. This particular bit will be disconnected and used as a spare after the program changeover.
- 4. Sustainer Engine Cutoff signal is generated by operation of the SECO Over-ride switch at the Spacecraft Communicator's Console. Operation of this switch inserts a "one" in bit position 57 and also into the three Abort Sequence Initiated bits, 53, 62, and 67.
- 5. This format constitutes a complete Telemetry Event Data Message. Transmission of this message is repeated each 74 milliseconds. The most significant bit is transmitted first.

3.16 LOG TAPE HIGH-SPEED OUTPUT PROGRAM (MXHSPR)

MXHSPR examines the log tape for high-speed output data transmitted to Cape Canaveral (DCC subchannel 3). This data is unpacked, scaled, and arranged for off-line printing. The flow chart for MXHSPR is shown in Figure 3-13.

3.16.1 Input Requirements

The only input to MXHSPR is the log tape of a Mercury mission—simulated or unsimulated—on tape unit B6. The log tape is described in subsection 3.15.1.

3.16.2 Output Requirements

The output tape, A5, contains the high-speed output data transmitted by Goddard to Cape Canaveral in tabular form. The data is segregated both by mission phase and by the displays serviced by the particular data.

When printed, each line of print from the output tape contains the logging time tag expressed in hours, minutes, and seconds of GMT. The remainder of the line depends upon the data and is a function of display equipment, code, mission phase, and the data frame.

The following examples of MXHSPR output show the computed values transmitted to Cape Canaveral to drive plotboards 1, 2, and 4 and the digital displays (during launch plotboard 3 is not drived from Goddard). MXHSPR searches the log tape for values transmitted to, for example, plotboard 1 at Cape Canaveral during the launch phase. These values are then assembled and printed under the heading LAUNCH PHASE—PLOTBOARD ONE as shown below.

LAUNCH PHASE - PLOTBOARD 1

	V/V _r		v/v_r
γ	(less than .19)	γ	(less than .9)
. 00000000	. 03999999	.00000000	.00000000
. 00000000	. 03999999	.00000000	.00000000
.00000000	. 0399999	.00000000	.00000000
.00000000	. 0399999	.00000000	.00000000
.00000000	. 0399999	.00000000	.00000000
.00000000	. 0399999	.00000000	.00000000

LAUNCH PHASE - PLOTBOARD 1 (Cont'd)

TIME	γ	$rac{ extsf{V/V}_{ extsf{r}}}{ extsf{(greater than .9)}}$
	99999809	. 89999998
	99999809	. 89999998
1	99999809	. 89999998
1	99999809	. 89999998
2	99999809	. 89999998
2	99999809	. 89999998

Plotboard 1 shows flight path angle γ and velocity ratio V/V_r associated with a TIME column. TIME is in seconds after liftoff. Because of plotboard scaling, three columns of both γ and V/V_r are shown.

The same operation is performed for the other plotboards and displays during the launch phase. MXHSPR also searches the log tape for the values displayed during the orbit phase. The values are assembled and printed under the heading ORBIT PHASE. Subheadings are printed for each of the plotboards, wall map, and digital displays as noted in the examples.

Some displays, for example plotboard 1, show different quantities during the launch and orbit phases. During launch, plotboard 1 shows flight path angle γ vs. velocity ratio (V/V_r); during orbit it shows altitude vs. velocity.

The example for plotboard 2 shows crossrange deviation (Y -Y $_{nom}$) and downrange distance D (D less than 60). TIME is in seconds after liftoff and additional columns of downrange distance, height H (D less than 60) are shown because of scaling.

LAUNCH PHASE — PLOTBOARD 2

Y-Y _{nom}	D (D less than 60)	H (D less than 60)	TIME	D (D greater than 60)	H (D greater than 60)
.05865091	.00000000	.00000000	1	.00000000	. 00000000
.05865091	.00000000	.00000000	1	.00000000	.00000000
.05865091	.00000000	.00000000	2	.00000000	.00000000
.05865091	.00000000	.00000000	2	.00000000	.00000000
.05865091	.00000000	.00000000	3	.00000000	.00000000

The first two columns of the plotboard 4 example show the latitude and longitude of the impact point during launch if the firing of the retrorockets was withheld until the spacecraft reached an altitude of just above 450,000 feet. The next two columns show the impact point if the escape rockets were fired immediately. Also, if tower separation has already taken place, then these columns show impact point if retrorockets are fired 30 seconds from present time. The column TIME indicates time in seconds associated with each set of values.

LAUNCH PHASE - PLOTBOARD 4

LATITUDE (maximum)	LONGITUDE (maximum)	LATITUDE (30 seconds)	LONGITUDE (30 seconds)	TIME
11.99999809	00586510	28.49266434	-80.50439835	
11.99999809	00586510	28.49266434	-80.50439835	
11.99999809	00586510	28.49266434	-80.50439835	1
11.99999809	00586510	28.49266434	-80.50439835	1
11.99999809	00586510	28.49266434	-80.50439835	2

The following example shows the values transmitted to the wall map at Cape Canaveral and the time in seconds after liftoff that the transmission occurred. It shows latitude, longitude, and time (present position of spacecraft.

LAUNCH PHASE - WALL MAP

LATITUDE (P. P)	LONGITUDE (P.P)	TIME
28.50439644	-80.41055679	
28.50439644	-80.41055679	
28.50439644	-80.41055679	1
28.50439644	-80.41055679	1
28.50439644	-80.41055679	2

Data transmitted to the strip charts (see the following example) is the result of computed values (as a result of input) versus nominal values. The first column is γ - γ_{nom} from B-GE data. The second column would be the same (γ - γ_{nom}) for either IP 7094 or raw radar data whichever is the selected source. The next two columns show the difference in velocity ratios $\left[\frac{V}{V_r} - \frac{V}{V_r}\right]$ (nominal) for B-GE and the IP 7094. The TIME column contains time in seconds after liftoff.

LAUNCH PHASE - STRIP CHARTS

γ-γ _{nom} (B-GE)	γ - γ _{nom} (AN/FPS-16)	$V/V_r - V/V_{r \text{ nom}}$ (B-GE	$V/V_r - V/V_{r \text{ nom}}$ (AN/FPS-16)	TIME
 00488663	00488663	 00003913	00003913	
 00488663	00488663	00003913	—. 00003913	
 00488663	 00488663	00003913	00003913	1
 00488663	00488663	00003913	00003913	1
 00488663	 00488663	00003913	00003913	2

The output of MXHSPR to some of the digital displays during launch is shown below. Following these examples is a glossary of the column headings:

LAUNCH PHASE - DIGITAL DISPLAYS

	Direction 1	- DIGITAL D	JIDI LIMID	
ΔT		Recovery Area	TI	ME
00 00	00	0 0		
00 00	00	0 0		1
00 00	00	0 0	:	1 2
00 00	00	0 0	;	2 3
00 00	00	0 0		3 4
r – R	γ	I.A.	O.C.	v/v _r
000.0	00.00	00.0	00	0.0000
000.0	00.00	00.0	00	0.0000
000.0	00.00	00.0	00	0.0000
000.0	02.30	28.3	00	0.0510
000.0	02.78	28.3	00	0.0511

 $\Delta \text{T}-\text{This}$ column indicates elapsed time to fire retrorockets to impact in next recovery area.

RECOVERY AREA—this column will contain two numbers:

00 - Recovery Area A

01 - Recovery Area B

02 - Recovery Area C

03 - Recovery Area D

04 - Recovery Area E

10 - Recovery Area 1A

11 - Recovery Area 1B

TIME - Time in seconds after liftoff:

 $r - \overline{R}$ - Height in miles

γ - Flightpath angle in degrees
I. A. - Inclination angle
O. C. - Orbit capability

V/V_r - Ratio of velocity to velocity required

Examples of values transmitted to Cape Canaveral during the orbit phase, logged on the log tape and subsequently printed by MXHSPR, are shown below. A glossary of unexplained column headings follows the examples.

ORBIT PHASE - WALL MAP

LATITUDE (P. P.)	LONGITUDE (PP)	LATITUDE (R. F. in 30 sec)	LONGITUDE (RF in 30 sec)	TIME
30.85043907	-69.85337257	03910065	17595291	5 30
30.92863846	-69.50146675	03910065	17595291	5 36
31.00684166	-69.14955902	03910065	17595291	5 42
31.08504295	-68.79765511	03910065	 17595291	5 48

ORBIT PHASE - DIGITAL DISPLAYS

	ODDIT				
GTRS	ORBIT NO.	GMTLC	LATITUDE	LONGITUDE	$r - \overline{R}$
04 28 02	01	20 02	19.36	-065.0	092.9
04 27 50	01	20 02	19.36	-065.0	092.9
04 27 36	01	20 02	19.36	-065.0	092.9
04 27 26	01	20 02	19.36	-065.0	092.9
04 27 14	01	20 02	19.36	-065.0	092.9

ORBIT PHASE - DIGITAL DISPLAYS

APOGEE HT.	I.A.	ORBIT CAP.	TIME	VELOCITY
124.9	32.4	00	5 36	25660
124.8	32.4	00	5 4 8	25660
124.8	32.4	00	6 2	25660
124.7	32.4	00	6 12	25660
124.7	32.4	00	6 24	25660
GMTRC	ECTRC	GMTRC EPO	ECTRC EPO	GMTRC-EOM
15 33 21	00 19 36	16 41 38	01 27 53	19 46 39
15 33 21	00 19 36	16 41 38	01 27 53	19 46 39
15 33 21	00 19 36	16 41 38	01 27 53	19 46 39
15 33 21	00 19 36	16 41 38	01 27 53	19 46 39
15 33 21	00 19 36	16 41 38	01 27 53	19 46 39
ECTRC-EOM	GMTRS	ICTRC	RECOVERY AR	EEA TIME
04 32 54	19 47 23	00 —01	16 1	1 6 6
04 32 54	19 47 23	00 -01	16 1	1 6 18
04 32 54	19 47 23	00 -01	16 1	1 6 30
04 32 54	19 47 23	00 -01	16 1	1 6 42
04 32 54	19 47 23	00 —01	16 1	1 6 54

ORBIT PHASE — PLOTBOARD 1

ALTITUDE	VELOCITY	TIME
92.86412510	25659.82404041	5 30
92.86412510	25659.82404041	5 36
92.86412510	25659.82404041	5 42
92.86412510	25659.82404041	5 48

ORBIT PHASE - PLOTBOARD 2

$r-\overline{R}$	TIME	ALTITUDE	TIME
109.09090826	334.31069946	93.25513181	5 30
109.09090826	334.31069946	93.25513181	5 36
109.09090826	334.31069946	93.25513181	5 42
109.09090826	351.90599918	93.25513181	5 48
109.09090826	351.90599918	93, 25513181	5 55

ORBIT PHASE - PLOTBOARD 3

PERIGEE LONGITUDE	ELAPSED TIME	ECCENTRICITY	TIME
-92.02346039	334.31069946	.00464514	5 30
-92.02346039	334.31069946	.00464514	5 36
-92. 02346039	334.31069946	.00436363	5 42
-91.31964874	351.90599918	.00436363	5 48
—91.31964874	351.90599918	.00436363	5 55

ORBIT PHASE - PLOTBOARD 4

LATITUDE (P. P.)	LONGITUDE (P. P.)	LATITUDE (I. P.)	LONGITUDE (I. P.)	TIME
30.86216545	-69.98544821	11.99999809	-45.00000191	5 30
30.95600700	-69.54545593	11.99999809	-45.00000191	5 36
31.02638817	-69.10557365	11.99999809	-45.00000191	5 42
31.09676933	-68.66569138	11.99999809	-45.00000191	5 48
31.19061089	-68.13783073	11.99999809	-45.00000191	5 55

A - semimajor axis of orbit of spacecraft

R - nominal radius of earth

Time — time is shown in seconds or minutes and seconds since liftoff

Perigee — lowest point of orbit

Eccentricity - eccentricity of orbital ellipse

Long. R. F. - longitude if retrofire in 30 seconds

GTRS — elapsed time to retro setting

GMTLC - GMT of landing (computed)

(r-R) - altitude

Apogee - farthest point of orbit

GMTRC - Greenwich Mean Time of retrofire computed

ECTRC - elapsed spacecraft time of retrofire computed

ECTRC-EPO - elapsed spacecraft time of retrofire computed, end present orbit

GMTRC-EOM - GMTRC at end of mission

GMTRS -Greenwich Mean Time of retro setting

ICTRC - incremental time of retrofire computed

3.16.3 Method

MXHSPR reads all data from the log tape, separates the high-speed output from the other data, and writes this on two auxiliary tapes thereby producing two identical high-speed data tapes. A search for the time of liftoff is made during the generation of these tapes. If no liftoff signal is found, MXHSPR assumes a liftoff time of zero. (There is no liftoff for log tapes produced by restarts or runs based on an orbit r, v.) The program then reads data from one of the tapes, unpacks the data pertaining to one of the displays, scales this data accordingly, and writes it on the output tape. When the logical end of the data tape is reached, the program rewinds the tape and immediately begins processing data for the next display by using the other data tape. This procedure continues until the data for each of the displays has been recorded on the output tape.

MXHSPR uses the Share program SE9OU2 to write data on the output tape.

3.16.4 Usage: Operator's Procedures

- a) Ready B6 with a log tape.
- b) Ready A5, B7, C6 with blank tapes.
- c) Ready card reader with MXHSPR absolute row binary deck.
- d) Press CLEAR and LOAD CARDS.
- e) Print A5 under program control.

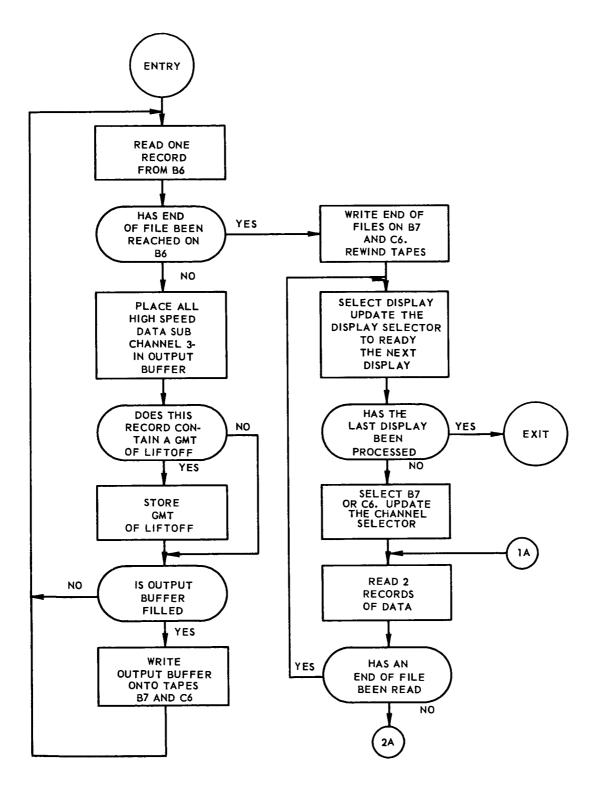


FIGURE 3-13. MXHSPR PROGRAM FLOW CHART (Sheet I of 2)

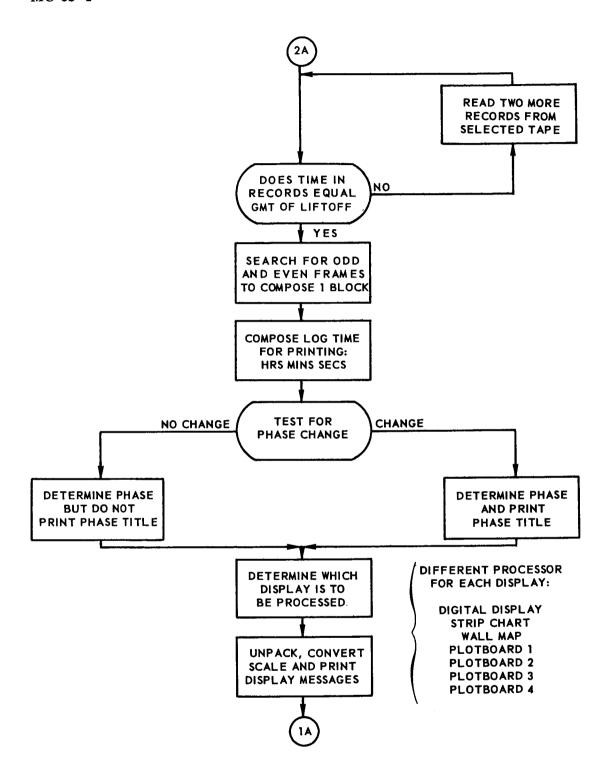


FIGURE 3-13. MXHSPR PROGRAM FLOW CHART (Sheet 2 of 2)

3.17 LOG TAPE PRINTER PROGRAM (MXPRLG)

MXPRLG extracts key-selected input and output teletype data from the log tape for off-line printing in a specified format.

The flow chart for MXPRLG is shown in Figure 3-14.

3.17.1 Input Requirements

Input to MXPRLG are log tapes of Mercury missions, including simulated runs. As needed, the tapes for a mission are mounted on tape units A6 and B6 as follows: the first reel, third, fifth, etc. use B6; the second reel, fourth, sixth etc. use A6. MXPRLG requires that the teletype subchannel(s) number be entered via Entry keys at the beginning of the run. If more than one log tape is to be processed, the number of tapes must also be entered via the keys.

3.17.2 Output Requirements

The final printed output will be on tape unit A2, with tape units B4 and B5 used as intermediate outputs. Low-speed messages are printed in BCD. The output tape A2 contains the selected messages in the format listed in Table 3-2.

In the input format example, Table 3-2, data was received through DCC subchannel 29. Each line following subchannel identification indicates a low-speed radar message in the format, as defined by the specifications. Each message contains the station identification code, the type of radar used (Verlort or AN/FPS-16), the time, range, azimuth, and elevation of the spacecraft position and whether the data is valid or invalid.

The first line of the output example, Table 3-2, contains the heading words IDENTIFICATION TERMINAL, and DATA IS TRANSMITTED. Immediately under the word IDENTIFICATION is a number corresponding to the DCC subchannel through which the data passed and was subsequently logged.

In the output format example, the data was transmitted through DCC subchannel 11. The left column contains the communications switch letters (YNYN); the GMT in hours, minutes, and seconds; the terminal station letters; and the type of transmission (AQ meaning acquisition). Following the type of transmission is the message content consisting of four "look" angles of time, range, azimuth and elevation.

	Table 3-2							
GODDARD TELETYPE INPUT								
Station Number		Validity	Hours	Minutes	Seconds	Azimuth	Elevation	Range
04	2	2	00	14	42	313663	001671	0432621
04	2	2	00	14	48	313622	002100	0421242
04	2	2	00	14	54	313621	002413	0407714
04	2	2	00	15	00	313537	002642	0376356
04	2	2	00	15	06	313526	003106	0365024
	GODDARD TELETYPE OUTPUT							
IDENTIFICATION TERMINAL DATA IS TRANSMITTED)					
	11							
YNYN 114800Z								
CY1 AQ								
11	54 3	1 1555	287.5	.8				
11	55 58	8 873	286.1	9.6				
11	57 04	4 382	280.8	30.4				
11	57 58	8 237	134.1	58.8				
114800Z CY1								

3.17.3 Method

The numbers of the subchannels and the total number of input tapes must be entered using the Entry keys. MXPRLG searches the log tapes for these subchannels and writes the data on the intermediate tapes B4 and B5. When the log tapes have been completely read, the B4 and B5 tapes are used as input and converted to the final printed output. Completed messages are written on A2.

MXPRLG uses the general purpose print program SE9OU2 and the teletype decoder program TYDC.

3.17.4 Usage

- a) Operators Procedure:
 - 1) Ready log tapes on A6 and B6
 - 2) Ready blank tapes on B4 and B5 for intermediate output
 - 3) Ready blank tape on A2 for final output
 - 4) Enter number of log tapes in keys 3 through 8, right-justified, and all subchannel numbers in keys 10 through 32
 - 5) Press CLEAR, then LOAD TAPE buttons

The machine will come to a halt (7₈) and will print the number of log tapes. If correct, press START to continue. If incorrect, check console keys and begin again.

- 6) MXPRLG will now run until all subchannels have been processed
- b) Error Conditions—if an error occurs in using the subroutine SE9OU2, the program transfers to a halt with 76400₈ in the address. In case of error in subroutine TYDC, MXPRLG prints on-line the message ERROR RETURN MESSAGE TOO LONG and continues to print out off-line that part of the message that has been preserved.

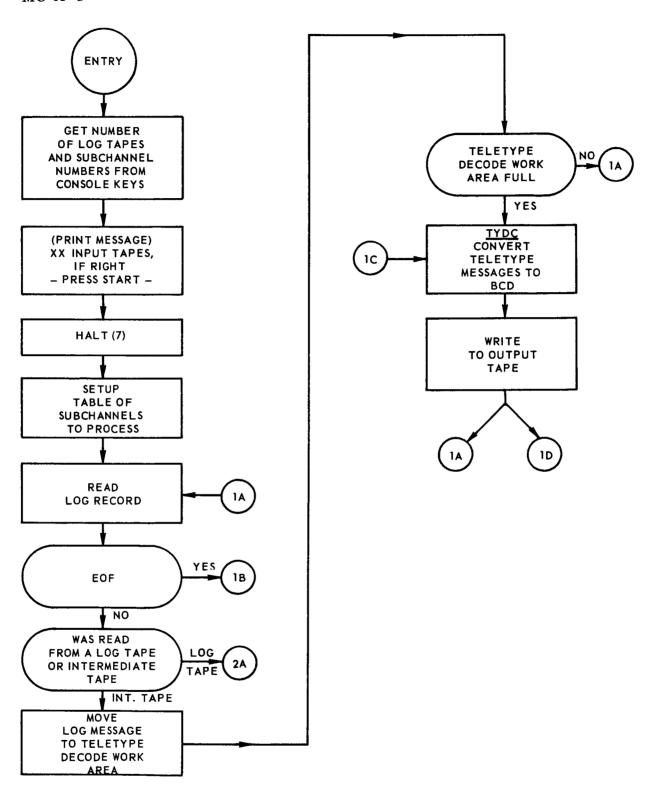


FIGURE 3-14. MXPRLG PROGRAM FLOW CHART (Sheet 1 of 2)

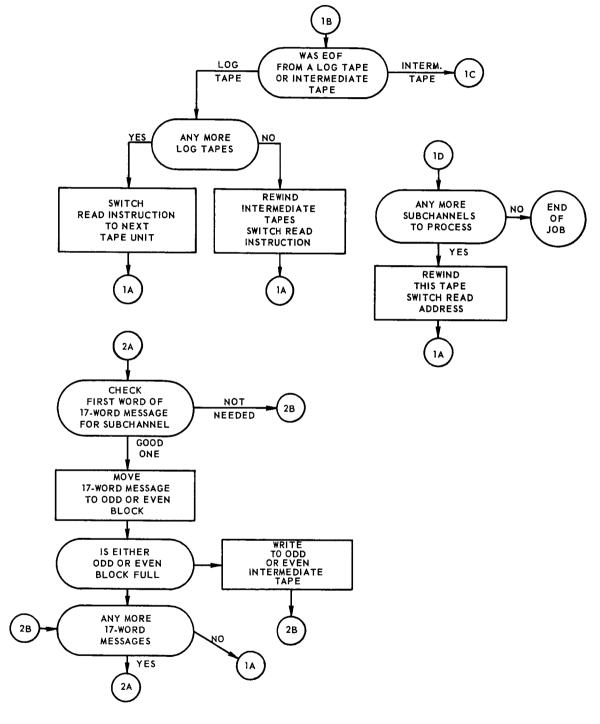


FIGURE 3-14. MXPRLG PROGRAM FLOW CHART (Sheet 2 of 2)

3.18 LOG TAPE PLOTTING PROGRAM (MXHSPL)

MXHSPL displays on the Goddard plotboard the data from the log tape used during a previous Mercury Programming System run to drive plotboards 1 through 4 and the wall map at the MCC.

The flow chart for the MXHSPL program is shown in Figure 3-15.

3.18.1 Input Requirements

The only input to MXHSPL is the log tape of a Mercury mission—simulated or unsimulated—on tape unit B6. (The log tape is described in subsection 5.1.1.) The plotboard number is entered in the console keys (the wall map is entered as 5).

3.18.2 Output Requirements

A plot is made for each flight phase on the X-Y plotter of those parts of the high-speed output messages sent over DCC subchannel 3 which refer to Cape plotboards or the wall map.

As each point is plotted, an on-line print indicates plotboard number and flight phase. After a complete phase has been plotted, the program stops and prints on-line, END OF ------ PHASE. CHANGE PAPER. HIT START.

3.18.3 Method

MXHSPL searches the log tape for high-speed output messages transmitted on subchannel 3 of the Data Communications Channel. When a complete message of 408 bits is found, the DCC is enabled and the bits referring to the display designated by the number in the console keys are packed in an output block to be sent to the DCC. A control word, which has a minus sign indicating how many arms of the plotter and how many pens per arm are to be used, is also packed in the DCC output block.

The message is transmitted through DCC subchannel 4 and the DCC is disabled. The remaining messages are located and processed until a phase change is found. The program then halts until the computer operator presses START. The program then processes the data for the next phase.

MXHSPL incorporates the Share program SE9OU2 as an internal subroutine for printing.

3.18.4 Usage

- a) Operator's Procedures:
 - 1) Ready the plotter and standard SOS system tapes
 - 2) Ready B6 with the log tape
 - 3) Ready the on-line printer
 - 4) Ready the card reader with the MXHSPL deck
 - 5) Enter the plotboard number in the console keys (enter wall map as 5)
 - 6) Press CLEAR and LOAD TAPE
 - 7) Program stops after each flight phase is plotted. To continue, press START after changing plotboard paper
- b) Error Conditions—an error return from the subroutine SE9OU2 results in a program halt.

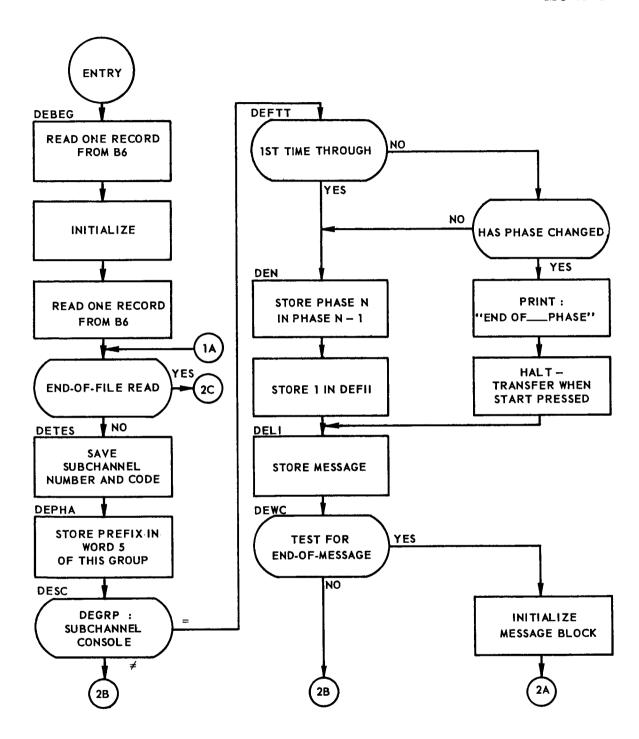


FIGURE 3-15. MXHSPL PROGRAM FLOW CHART (Sheet 1 of 2)

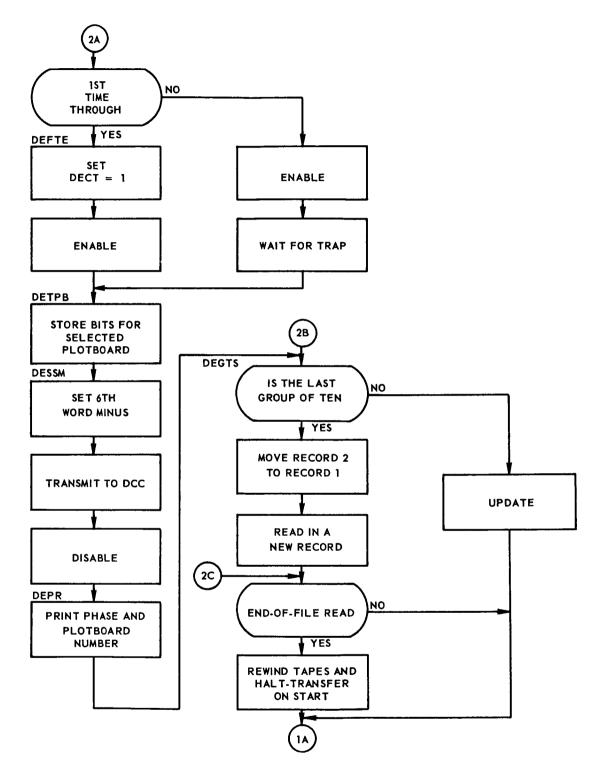


FIGURE 3-15. MXHSPL PROGRAM FLOW CHART (Sheet 2 of 2)

SECTION 4

SUPPORTING PROGRAMS

Mercury System supporting programs perform all functions necessary for object program execution that cannot be performed during the mission. Basically, supporting programs produce and debug Mercury system tapes. Since they are not part of the real-time tracking system, they are used either before or after each real-time operation.

Message and station characteristics tapes are produced by the MXWMOT, U0STCH, and U0STUP programs; the B4 utility tape, by the WRTB4T and HOMER programs. This utility tape contains the writer, loader, and dump programs. An input tape for the SOS compiler is prepared by MXMRGE. SOS output is handled by writer and loader programs MXSTWI, MXDEFN, MXLOAD, and SETORG which produce the absolute Mercury system tapes and load them into the machine.

The MTTEST transfer test processor serves as a debugging aid, and is generally run in simulated real-time.

A symbolic dump of core storage is accommodated with the programs SGENDX, ISODMP, and CORING used in conjunction with SOS.

4.1 MONITOR MERGE PROGRAM (MXMRGE)

In normal SOS, a compiled squoze deck becomes a job deck with the addition of three or more symbolic (Hollerith coded) control cards—the first card added is always a JOB card, and the last, a PAUSE card. This job deck may be entered into the computer for execution indirectly, via tape, or directly, via the on-line card reader. Further modifications in the program can be made with symbolic cards in combination with the original squoze deck.

The squoze deck of a complex system may become too large to handle conveniently. For this reason the squoze output from the Mercury SOS compilation was put on tape rather than cards. With MXMRGE, the symbolic cards are merged into this "squoze deck on tape" to produce a job tape as output. (The job tape is, in fact, identical to the tape produced when a job deck—squoze deck plus symbolic control—is written on tape off-line to be used as input to the computer.)

A general flow chart and a detailed flow chart for the MXMRGE program are shown in Figures 4-1 and 4-2.

4.1.1 Input Requirements

Input to MXMRGE consists of:

- a) The squoze deck written in column binary form on tape B8. This squoze tape must contain at least one record preceding and one record following a blank record (a minimum total of three records) and an end-of-file mark following the last record.
- b) Symbolic control cards read into the computer either directly, from the on-line reader, or indirectly, from tape A5.

4.1.2 Output Requirements

The output from this program consists of one or more merged jobs written on tape A3. For each job completed, the computer prints out the number of squoze records read from B8 and the number of modification cards (MOD to ENDMOD, exclusive) merged, with the message: GOOD MORNING. RESET ENTRY KEYS FOR FIRST JOB. THEN PRESS START.

4.1.3 Method

a) Symbolic cards are read by the RCD subroutine (internal to MXMRGE) which converts the cards to BCD and stores them in 12 consecutive

core storage cells. The proper leading address is placed in the thirteenth and fourteenth cells; the 14-cell block is written on tape A3 in BCD.

- b) With the exception of the blank record, the records of the squoze tape are read in one by one; each record is placed in consecutive cells of a 26-cell block, the unused cells set to zero. The proper leading address is placed in the twenty-seventh and twenty-eighth cells, and the 28-cell block is written on tape A3 in BCD.
- c) The blank record of the squoze tape is converted to BCD by placing Hollerith blanks in 12 consecutive core locations; the proper leading address is placed in the thirteenth and fourteenth cells, and the 14-cell block is written on tape A3 in BCD.
- d) When a PAUSE card is sensed by the card reader (or an end-of-file mark if tape A5 is used for the symbolic control card), an end-of-file mark is written on tape A3.

4.1.4 Usage

- a) Operator Procedures—usually a MXMRGE run immediately precedes an SOS run to process the merged jobs. Both phases require the standard SOS tapes. The operator must:
 - 1) Ready the following tapes:

	For MXMRGE	After MXMRGE, for processing*
A1	SOS tape	SOS tape
A2	(Not used)	BCD output
A3	Pool tape; used by MXMRGE to write the job tape	Job tape from MXMRGE
В1	Pool tape	Pool tape
B2	Pool tape	Pool tape
B8	Mercury System tape	(Not used)

^{*}Other tapes may be needed by the program being processed.

Ready the card reader with: 2)

WDBL2

Two cards

MXMRGE

40 cards

Job Decks

Not needed if they are being read from tape A5.

3) Set console entry keys S, 1, 2, 3, 4 and 5 as follows:

ENK-S up

Control cards read from A5.

ENK-S down

Control cards read from card reader.

ENK-1 up

B8 is not rewound after each input job.

ENK-1 down

B8 is rewound after each input job.

ENK-2 up

JOB card appears on squoze input.

ENK-2 down

No JOB card on squoze input. *

ENK-3 up ENK-3 down A5 is not rewound at end of MXMRGE.

A5 is rewound at end of MXMRGE.

(Note: if ENK-S is up, ENK-3 has no effect and MXMRGE rewinds A5 before and after all merging.)

ENK-4 up

One end-of-file mark between each squoze deck. *

ENK-4 down

Two end-of-file marks between each squoze deck.

ENK-5 down

ENK-5 up

Job deck and squoze deck with same characters in first card, columns 16 and 17, are merged.

No job select; job deck merged with next squoze

on tape.

- Ready the on-line printer; press CLEAR and LOAD CARDS. 4)
- Six lines are normally printed for each job, one line at the start of the job and five lines at the end of the job:

Line 1: JOB card

Line 2: ONE JOB HAS BEEN WRITTEN ON A3. THE SQUOZE

AND MOD COUNTS WILL FOLLOW.

Line 3: THE FOLLOWING 36 BIT BIN. NO. IS THE NO. OF THE

SQUOZE RECORDS READ FROM B8.

^{*}Applies to an obsolete version of SOS. ENK-2 and ENK-4 should always be up.

Line 4: (36-bit number)

Line 5: THE FOLLOWING 36 BIT BIN. NO. IS THE NO. OF MOD CARDS READ MOD TO ENDMOD.

Line 6: (36-bit number)

- b) Error Conditions—there are six possible error stops; each prints a specific line on the printer:
 - 1) REDUNDANCY ON CH. A. PRESS START IF COMPLETION OF MXMRGE DESIRED.
 - 2) REDUNDANCY ON CH. B. PRESS START IF COMPLETION OF MXMRGE DESIRED.
 - 3) REDUNDANCY ON CH. A. MXMRGE CANNOT COMPLETE MERGE.
 - 4) ILLEGAL HOLLERITH CHARACTER DETECTED BY RCD SUB-ROUTINE.
 - 5) FIRST CARD FOR JOB NOT JOB CARD—ERROR IN HOLLERITH CONTROL DECK.
 - 6) FALSE END-OF-FILE. INCORRECT SETUP OF HOLLERITH CONTROL DECK.

The first two errors can be bypassed and MXMRGE completed. The last four errors cannot be bypassed. If the error occurred on the first job, correct the condition, if possible, and start over. If the error occurred after the first job, either rerun all the jobs or remove A3 and start over with the uncompleted jobs.

c) Example of MXMRGE Usage—OUTPUT is the result of merging IN-PUT 1 and INPUT 2 and is written on A3.

INPUT 1	INPUT 2	OUTPUT
		<u>List Job Deck</u>
Squoze Tape	JOB LS Blank card PAUSE	JOB (BCD) LS (BCD) Squoze deck preceding blank of squoze (CB) Blank card of squoze deck (BCD) Squoze deck following blank of squoze (CB) Blank card (BCD) PAUSE (BCD) End-of-file mark

INPUT 2 INPUT 1 **OUTPUT** Execution Job Deck Squoze Tape JOB JOB (BCD) LG MOD Squoze deck preceding blank of squoze (CB) *Modification cards MOD (BCD) **ENDMOD** Blank card Modification cards (BCD)—present only if placed in card reader or on A5. *Data sentence deck ENDMOD (BCD) GO Blank of squoze deck (BCD) PAUSE Squoze deck following blank of squoze (CB) Blank card (BCD) Data sentence decks (BCD)-present only if placed in card reader or on A5 GO (BCD) PAUSE (BCD) End-of-file mark List Squoze Deck Squoze Tape JOB JOB (BCD)

LG LG (BCD) MOD Squoze deck preceding blank of squoze *Modification cards MOD (BCD) Modification cards (BCD)-present if placed **ENDMOD** in card reader or on A5 Blank card LIST ENDMOD (BCD) Blank card Blank of squoze (BCD) PAUSE Squoze deck following blank of squoze (CB) Blank card (BCD) LIST (BCD) Blank card (BCD) PAUSE (BCD)

Punch New Squoze Deck

End-of-file mark

Squoze Tape JOB JOB (BCD)
PS PS (BCD)
MOD Squoze deck preceding blank of squoze (CB)

^{*}if needed

INPUT 1

INPUT 2

OUTPUT

Punch New Squoze Deck—Continued

*Modification cards

MOD (BCD) Modification cards (BCD)

ENDMOD Blank card

ENDMOD (BCD)

PAUSE

Blank for squoze (BCD)

Squoze deck following blank of squoze (CB)

Blank card (BCD) PAUSE (BCD)

End-of-file mark

Punch Absolute Binary

Squoze Tape

JOB PA

JOB (BCD) PA (BCD)

MOD

Squoze deck preceding blank of squoze (CB)

*Modification cards

MOD (BCD)

ENDMOD Blank card

Modification cards (BCD)—present if

placed in card reader or on A5

PAUSE

ENDMOD (BCD)

Blank of squoze deck (BCD)

Squoze deck following blank of squoze (CB)

Blank card (BCD) PAUSE (BCD) End-of-file mark

A FIELD card may be placed at the beginning of a MXMERGE d) deck. The address of the FIELD card specifies that all mods with alter numbers lying in the range of the field will be automatically inserted in each mod packet. If the job is loaded on-line, a blank A5 is required. Column 72 of the job card must contain the job number $(1, 2, \ldots, 9)$. Example:

Exam	ple:

FIELD	1,500	Column 72
JOB LG MOD	ONE	1
ALTER CLA	315,317 SMTNG	
ENDMOD GO JOB	TWO	2

^{*}if needed

FIELD	1,500	Column 72
LG		
MOD		
ALTER	600	
ADD	OTHR	
ALTER	15	
\mathtt{STL}	LSWR	
ENDMOD		
GO		
PAUSE		

This MXMRGE deck effectively becomes:

JOB LG	ONE	1
MOD		
ALTER	315,317	
CLA	SMTNG	
ALTER	15	
\mathtt{STL}	LSWR	
ENDMOD		
GO		
JOB	TWO	2
LG		
MOD		
ALTER	315,317	
CLA	SMTNG	
ALTER	15	
\mathtt{STL}	LSWR	
ALTER	600	
ADD	OTHR	
ENDMOD		
GO		
PAUSE		

e) A PAUSE card is no longer required between successive jobs.

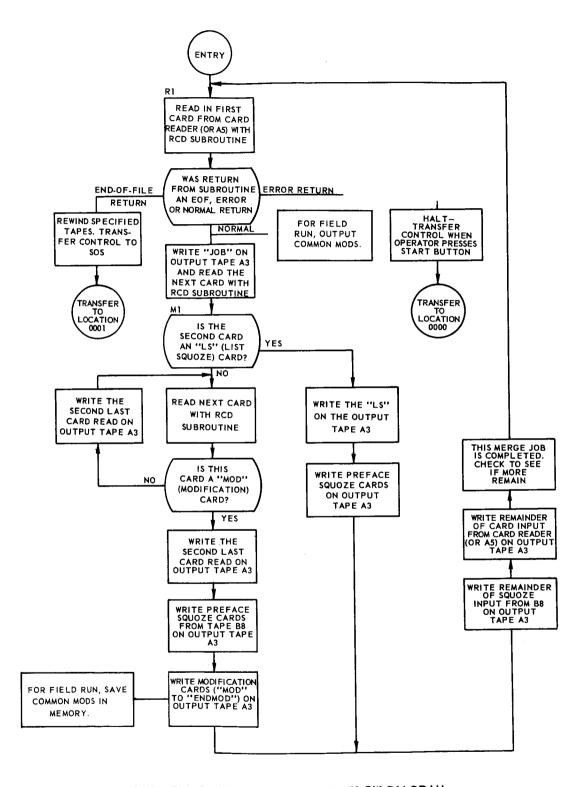


FIGURE 4-1. MXMRGE GENERAL FLOW DIAGRAM

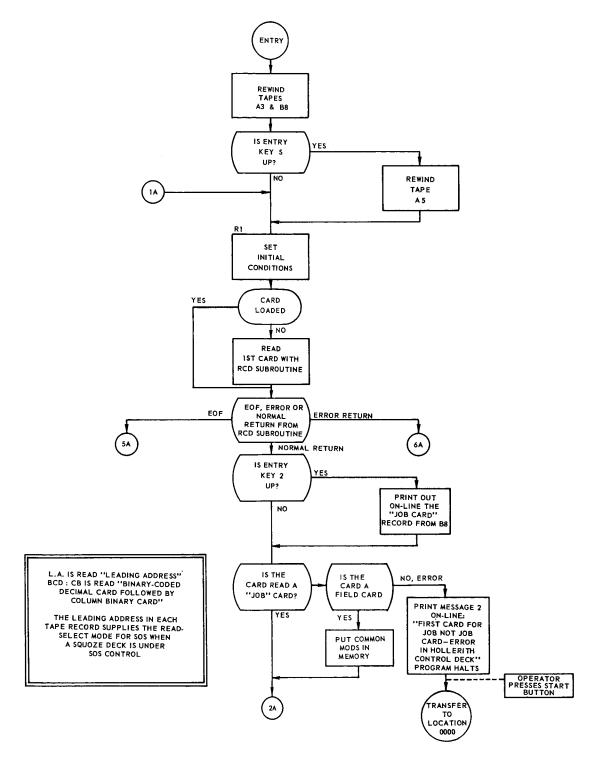


FIGURE 4-2. MXMRGE PROGRAM FLOW CHART (Sheet 1 of 6)

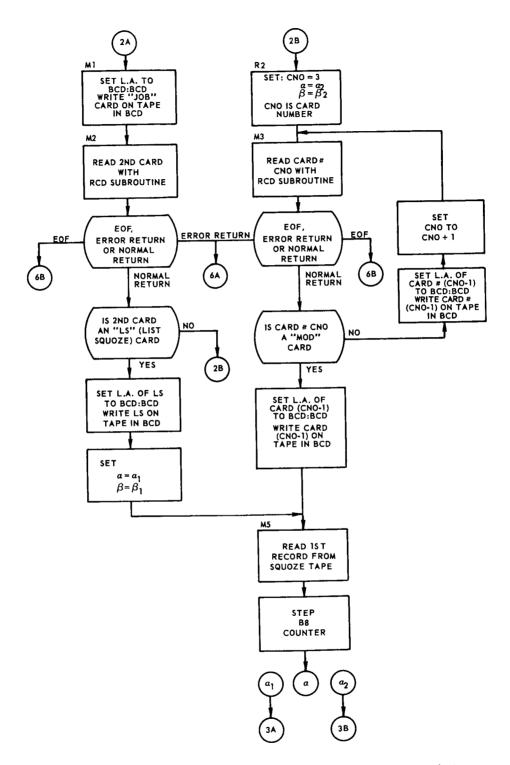


FIGURE 4-2. MXMRGE PROGRAM FLOW CHART (Sheet 2 of 6)

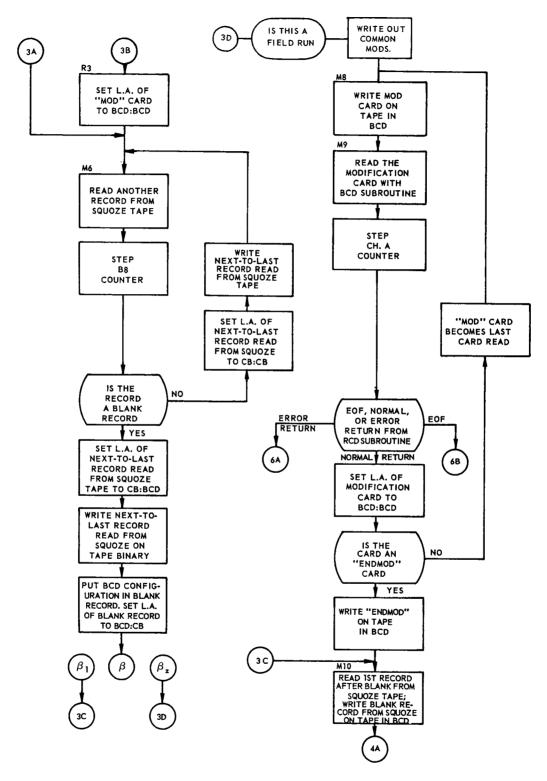


FIGURE 4-2. MXMRGE PROGRAM FLOW CHART (Sheet 3 of 6)

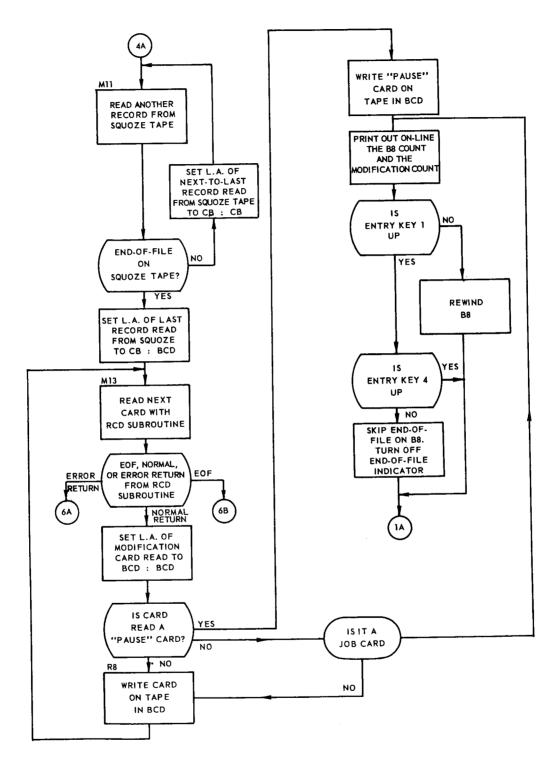


FIGURE 4-2. MXMRGE PROGRAM FLOW CHART (Sheet 4 of 6)

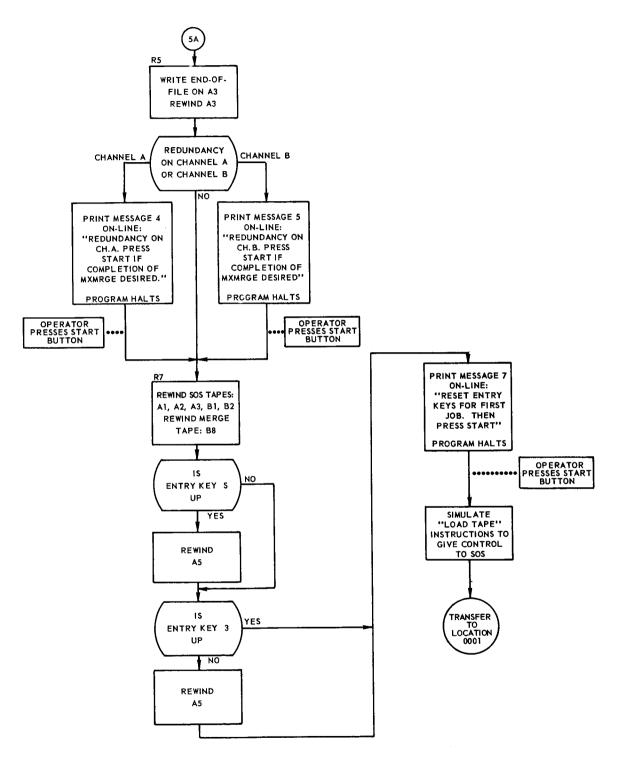


FIGURE 4-2. MXMRGE PROGRAM FLOW CHART (Sheet 5 of 6)

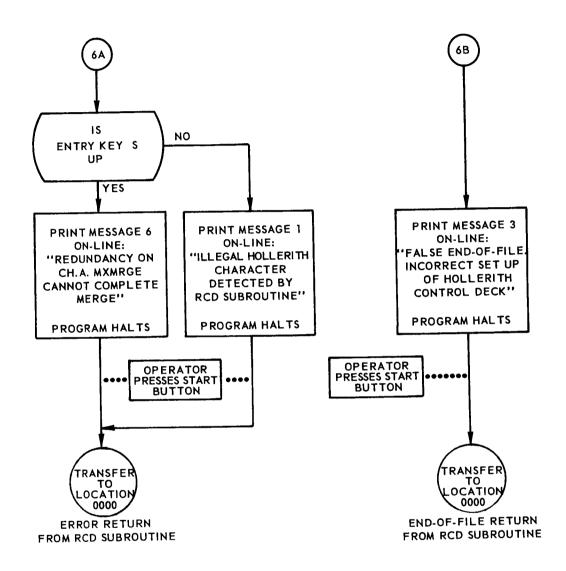


FIGURE 4-2. MXMRGE PROGRAM FLOW CHART (Sheet 6 of 6)

4.2 MERCURY SYSTEM TAPE WRITER PROGRAM (MXSTW1)

MXSTW1 writes in absolute binary the real-time Mercury operational programming system onto one, two, or three self-loading system tape(s).

The flow chart for MXSTW1 and its subroutines RDWRT, RDUNCY, and CMPAR is shown in Figure 4-3.

4.2.1 Input Requirements

Input to MXSTW1 consists of intermediate tapes B1 and B3, produced by an SOS compilation, which contains the product of a squoze tape and mod deck merging operation. Also, on entering MXSTW1, the AC will contain the origin and length of BCOMB, a table of symbol definitions. This table lists the values to be substituted for symbols used in writer and loader instructions which reference system locations. The values are contained in the address portion of table entries.

4.2.2 Output Requirements

Output from MXSTW1 is the absolute binary, self-loading Mercury operational system tape on A6.

4.2.3 Method

SOS reads in the A3 job tape and writes the system mediary input tape (SYSMIT) on B1. B1 will contain Job 1 in binary and the dictionary of program symbols. SOS then loads SYSMIT from B1 into memory and transfers control (TCD) to SETORG. SETORG modifies SOS monitor communication cells to permit communication between Jobs 1 and 2; then transfers control to MXSTW1. The second time through MXSTW1, SOS writes SYSMIT for Job 2 on the B3 tape and the process cited for Job 1 is repeated for Job 2.

MXSTW1 first checks the Entry keys to determine the number of system tapes to be written and then tests for the job number—JOB 1 or JOB 2. JOB 1 is loaded the first time through, and MXSTW1 writes the self-loading file (MXLOAD) as the first file and the rest of the files from B1 on the first part of the A6 tape. MXSTW1 then writes the communication record (QDEFN) to be read when JOB 2 is written. This record contains the job number, the number of files written on the system tape(s) in JOB 1, and redundancies which occurred in JOB 1.

After all files from JOB 1 are written, MXSTW1 changes the SYSMIT tape from B1 to B3 and transfers control to SOS. SOS then writes SYSMIT for JOB 2

on B3, loads it into memory, and transfers to MXDEFN. The procedure is the same as for JOB 1, except MXLOAD is not written as the first file on JOB 2. Instead, the communication record is read. After the information is obtained from the communication record, all files on JOB 2 are written. This completes the writing of the A6 tape, and the writer duplicates SYSMIT B1 and B3 to retain the dictionary and other information for dumps. MXSTW1 then uses a subroutine which compares the A6, B8, and C6 tapes (if all three are used). After the tapes are compared, A6 is dialed to A1, the tapes are rewound, and the LOAD TAPE button is pressed.

4.2.4 Usage

Operator's Procedures:

- a) Ready the standard SOS system tapes, using A3 as the input tape.
- b) Ready the on-line printer.
- c) Press CLEAR and LOAD TAPE buttons.
- d) Enter in the keys the number of the system tapes to be written.
- e) Ready the following tapes depending on the number of system tapes to be written:
 - 1) A6—if one tape is desired
 - 2) A6 and B8—if two tapes are desired
 - 3) A6, B8, and C6-if three tapes are desired
- f) After each SYSMIT is loaded, control is transferred to MXDEFN and then to MXSTW1. The program halts after printing an on-line message to indicate the control transfer.
- g) Press START to produce the self-loading Mercury System tape(s).
- h) Any tape redundancies occurring during the writing of the system tape(s) are indicated by an on-line message which indicates where the redundancy occurred and what action to take to continue writing the tape(s).
- i) MXSTW1 provides an on-line printout stating that the Mercury System tape(s) is successfully written, and if more than one is written, START should be pressed to compare the tapes.

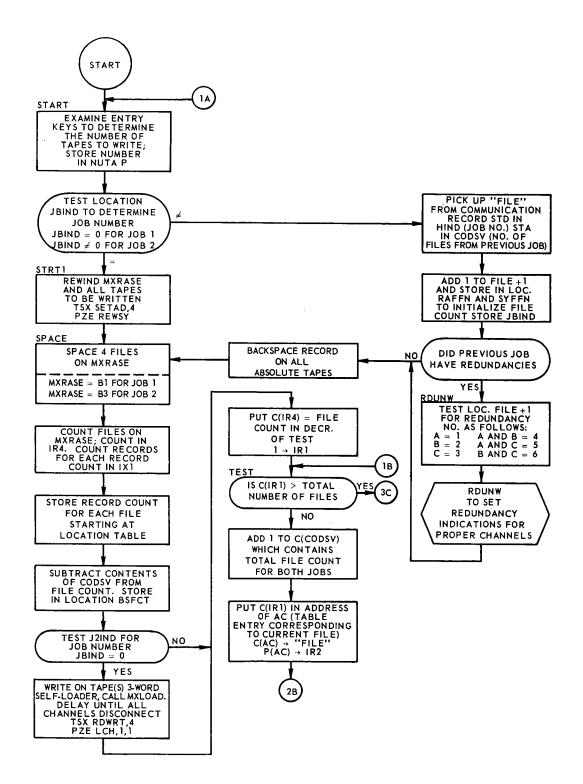


FIGURE 4-3. MXSTW1 PROGRAM FLOW CHART (Sheet 1 of 9)

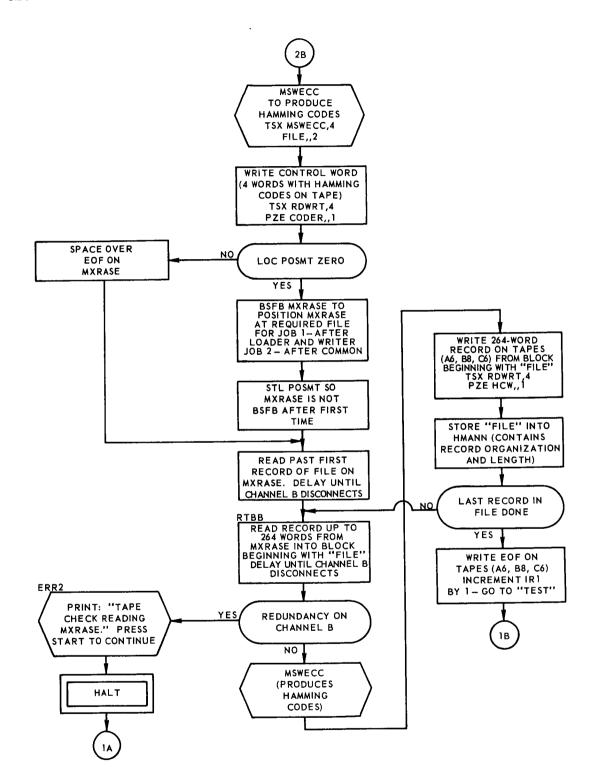


FIGURE 4-3. MXSTW1 PROGRAM FLOW CHART (Sheet 2 of 9)

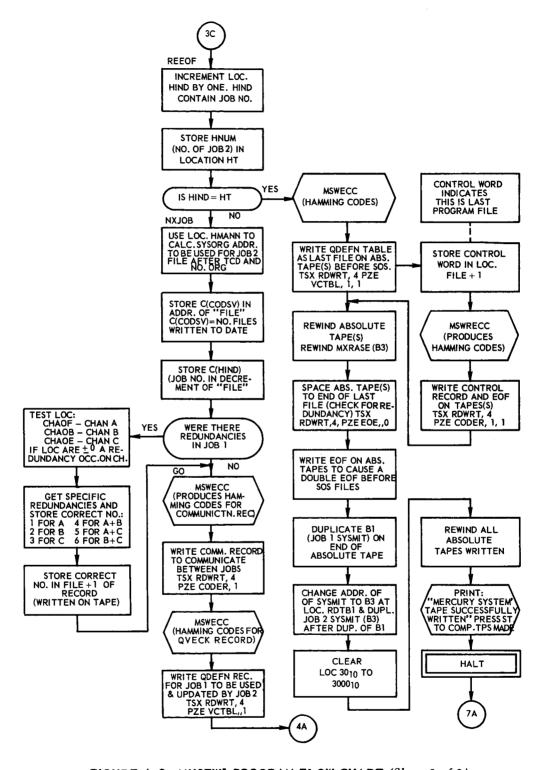


FIGURE 4-3. MXSTW1 PROGRAM FLOW CHART (Sheet 3 of 9)

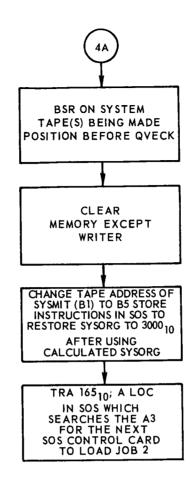


FIGURE 4-3. MXSTW1 PROGRAM FLOW CHART (Sheet 4 of 9)

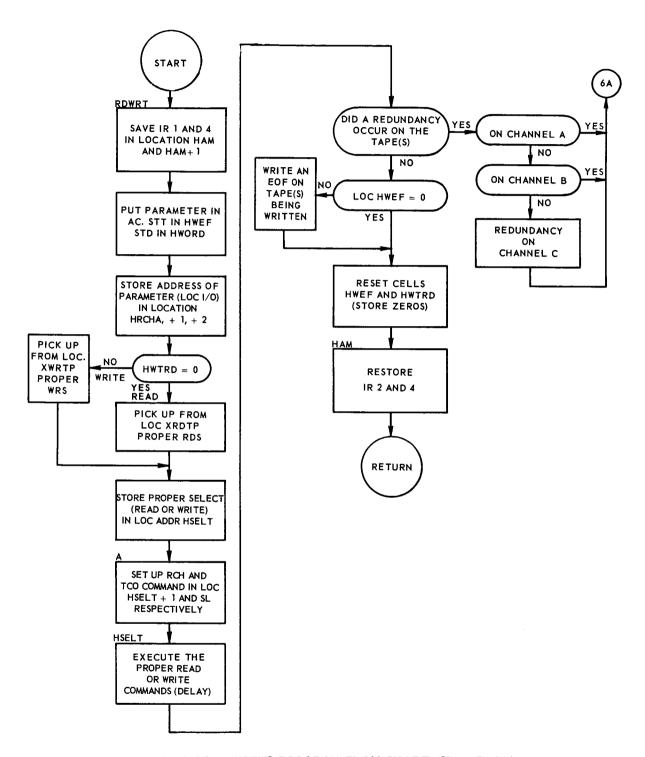


FIGURE 4-3. MXSTW1 PROGRAM FLOW CHART (Sheet 5 of 9)

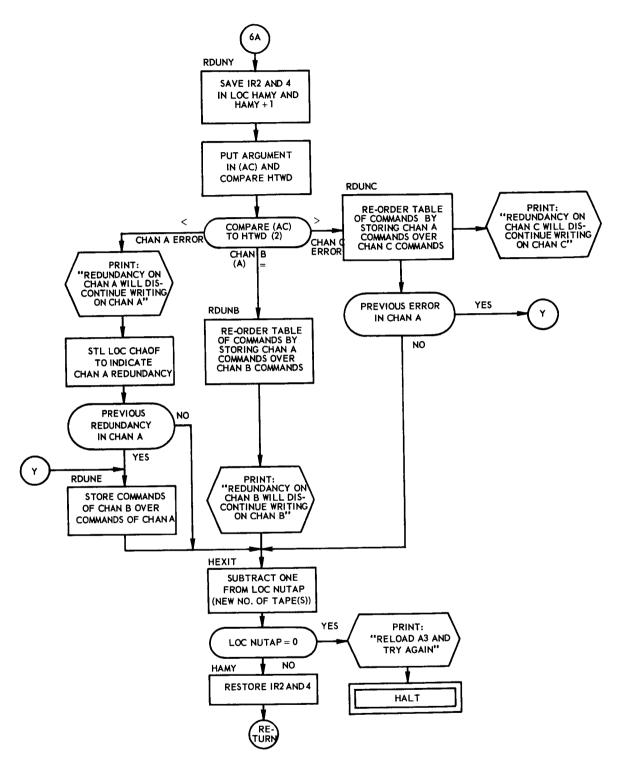


FIGURE 4-3. MXSTW1 PROGRAM FLOW CHART (Sheet 6 of 9)

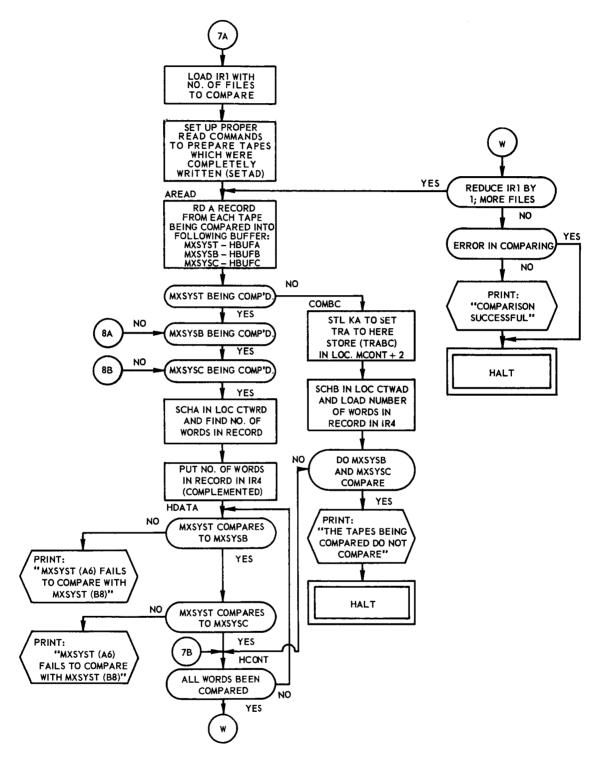


FIGURE 4-3. MXSTW1 PROGRAM FLOW CHART (Sheet 7 of 9)

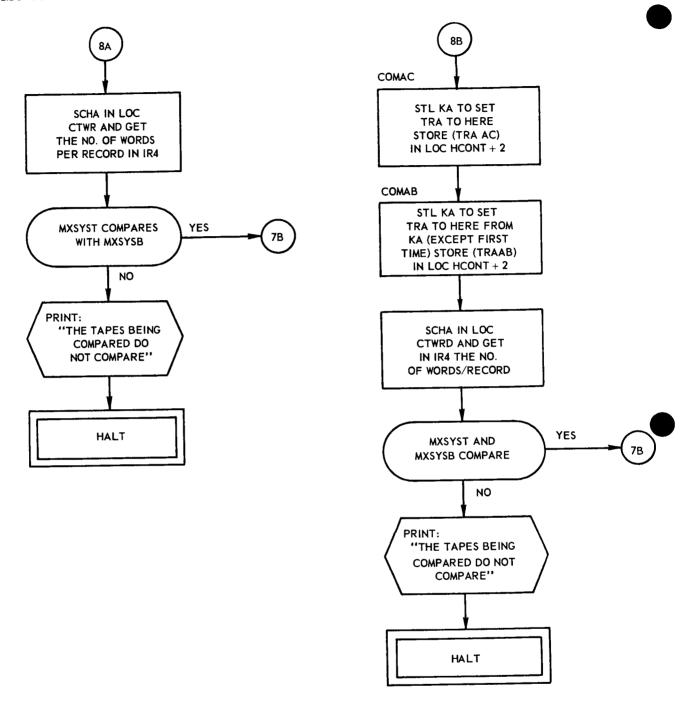


FIGURE 4-3. MXSTW1 PROGRAM FLOW CHART (Sheet 8 of 9)

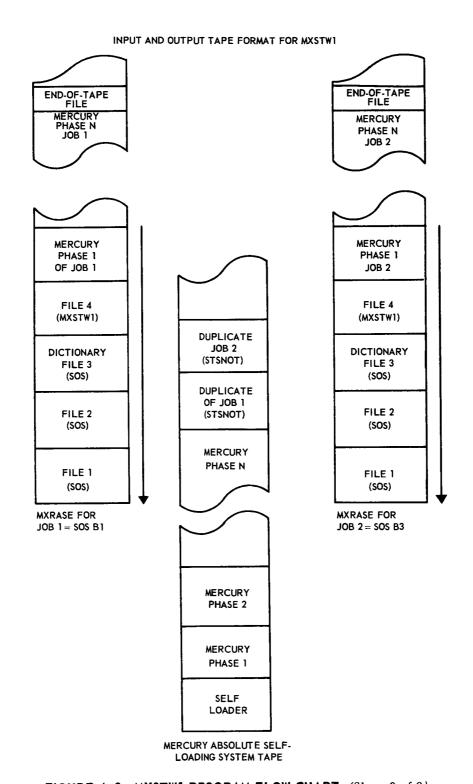


FIGURE 4-3. MXSTW1 PROGRAM FLOW CHART (Sheet 9 of 9)

4.3 MERCURY SYSTEM TAPE LOADER (MXLOAD)

MXLOAD occupies the first file of the self-loading Mercury operational system tape (A1) and loads into core the programs and tables needed by the system for launch data processing.

The flow chart for MXLOAD is shown in Figure 4-4.

4.3.1 Input Requirements

- a) MXLOAD as the first file on A1 and, if used, auxiliary tapes B8 and C6.
- b) The Mercury system program files, which are error-corrected with hamming codes and contain:
 - 1) The file number in the address and the record count in the decrement of the first data word in the first record of each file
 - 2) A specially formatted information record in the second record of each file. This record contains information for loading, table processing, and coring. The format for this record is given in the write-up for MYRSYS (see MC 63-2)
- c) The TMDEFN table, an error-corrected file immediately following the last program file.
- d) Two preset tables referenced by MXLOAD:
 - 1) TMHBUF—origined within the "common" area. TMHBUF entries have the format

PZE D, E, F

where D is the length of a buffered routine, E is the number of routines in the buffer block to be reserved in B unit, and F is the block number for this block reserved in B

2) TMBF00—set at the end of COMMON in JOB 2 and origined into the TMBF00 buffer. The first two words of each 3-word subset are preset:

First word—PZE E,1,TMBFXX+E Second word—PZE A length,, A ORG Third word—PZE 0 First word: P is negative the first time the buffer is used

D = E, the number of entries in TMBFXX

T must be 1

A = location of TMBFXX+E

The second word defines the A buffer. The third word defines the B buffer.

4.3.2 Output Requirements

When MXLOAD has been executed, the programs of the Mercury Real-Time Programming System, which are needed initially, are set for initial conditions and stored in memory.

The following tables, which are either completely built by MXLOAD or partially preset and built by MXLOAD, will be in memory. Only the entries which concern MXLOAD are given below:

- a) TMCORE—contains a 3-word subset for every file on the absolute system tape:
 - 1) First word—MZE indicates that the file is in A bank of core; PZE indicates that the file is not in A bank. The decrement contains the last location + 1 of the file, the tag contains the JOB number, and the address contains the first location of the file
 - 2) Second word—a 6-character BCD
 - 3) Third word—address contains the B origin of the block for the buffered routines
- b) TMQKY2-address is set with the TMCORE entry for the routine.
- c) TMREFR—prefix is set minus. The decrement is set with the file number for all routines except buffered routines which are in core. The decrements for these routines are set with the buffer number.
- d) TMBF00—the third word in each subset is set. The decrement contains the B buffer length and the address contains the B origin of the buffer.

- e) TMBFXX—contains a 3-word subset for every buffered routine. The first word of every subset is set by MXLOAD. The address has the routine number and the decrement contains the routine length. The prefix contains the location of the routine.
 - If P is negative, the routine is in A core. If bits 1, 2 = 01, the routine was last at the lower part of B block. If bits 1, 2 = 10, the routine was last at the upper part of B block.
 - 2) If P is positive, the routine is in B core. Bits 1, 2 have the same significance as above. If bits 1, 2 are blank, the routine is on tape.

When all the programs and conditions have been set, MXLOAD transfers control to the real-time program through M0INIT.

4.3.3 Method

When LOAD TAPE is pressed, a 3-word sequence bootstraps MXLOAD into memory and transfers control to MXLOAD.

MXLOAD reads the first five files into memory and passes control to MKTBL, a subroutine which extracts and processes information from TMHBUF for entries to the internal loader tables and TMBF00. MKTBL makes the following entries in the indicated tables:

- a) BLOK1—address contains the origin of the block, tag contains the number of routines in the file.
- b) BORGN-address contains the origin of the block.
- c) TABLE—address contains the block length, tag contains number of routines in the block, decrement contains the block number.
- d) TMBF00—the third entry of a subset contains the block length in the decrement and the block origin in the address.

MKTBL then returns control to MXLOAD. MXLOAD continues to read the program files by transferring to the MSLOAD subroutine with the requested file number in the calling sequence. MSLOAD reads and decodes the first record, then tests this information to see if it is the last program file. If it is not the last program file, MSLOAD tests for the requested file and positions the tape to the correct file. Information taken from this record is used to read in the proper number of records from this file.

The second record—the information record—is read in and decoded by the HOME 1 subroutine. Information is extracted from this record to make entries in TMCORE, TMREFR, and TMQKY2. The information is then tested for one of six possible ways to process the record.

The file is:

- a) Loaded into A core.
- b) Loaded into A core and buffered.
- c) Loaded into B core and executed.
- d) Loaded into a block in B core.
- e) Left on tape to be loaded into A core.
- f) Left on tape to be loaded into A core and buffered.

After all program files have been loaded, the TMDEFN file is loaded and control is transferred to MOINIT.

4.3.4 Usage

- a) The Mercury absolute system tape is mounted on A1 with the optional tapes, if used, mounted on B8 and C6. The machine should be in the 65K storage and multiple-tag modes. The ECC and ICC are set for execution and to reference A core.
- b) When LOAD TAPE is pressed, MXLOAD is bootstrapped in and control is transferred to MXLOAD. After MXLOAD has been executed, control is transferred to M0INIT. MXLOAD origins at location 1008; the last location is approximately 23528.
- c) If a redundancy occurs, an on-line message will state the two options:
 - 1) If sense key 4 is down, the tape will back up one record and try to reload when START is pressed
 - 2) If sense key 4 is up, the remaining program files will be loaded from the optional tapes, B8 and/or C6, when START is pressed.

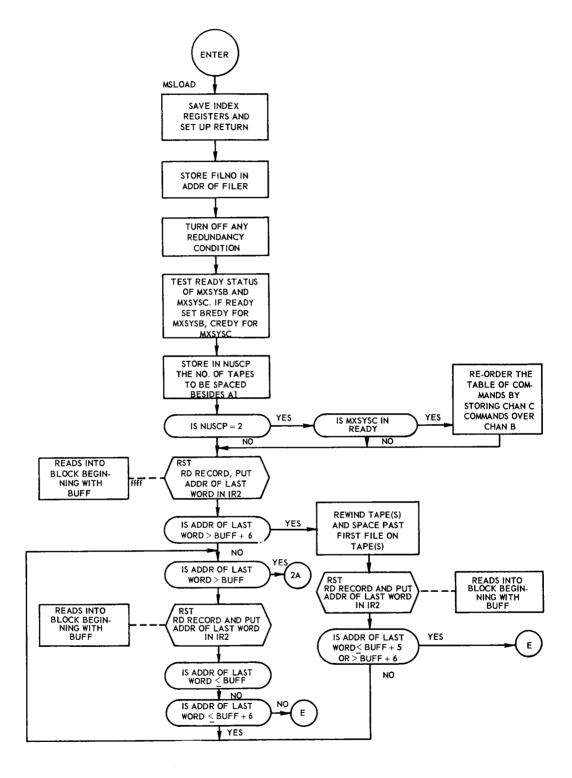


FIGURE 4-4. MXLOAD PROGRAM FLOW CHART (Sheet 1 of 14)

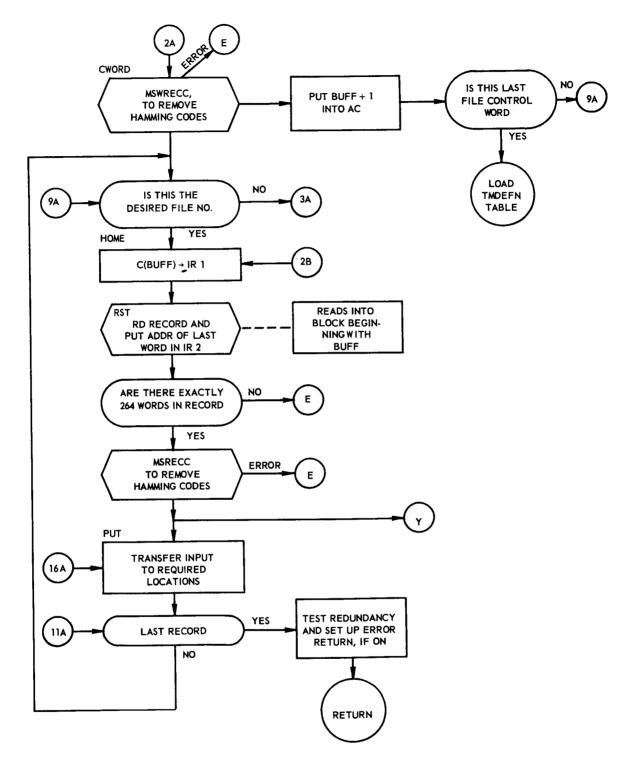


FIGURE 4-4. MXLOAD PROGRAM FLOW CHART (Sheet 2 of 14)

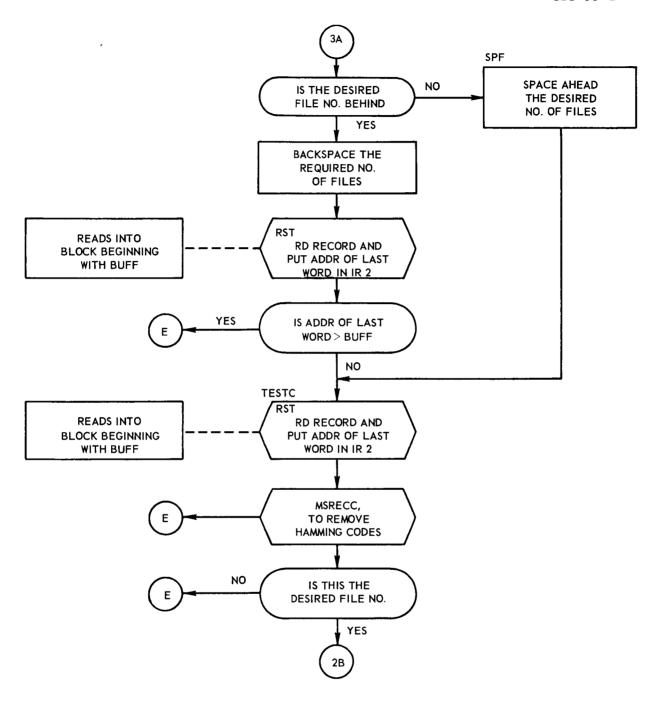


FIGURE 4-4. MXLOAD PROGRAM FLOW CHART (Sheet 3 of 14)

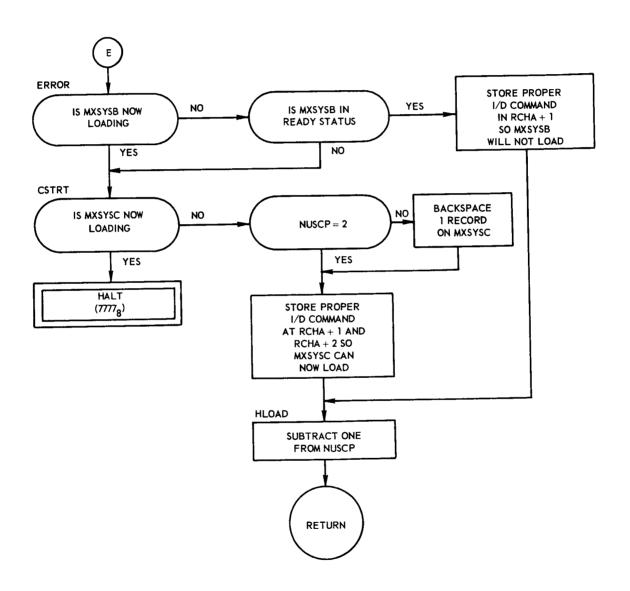


FIGURE 4.4. MXLOAD PROGRAM FLOW CHART (Sheet 4 of 14)

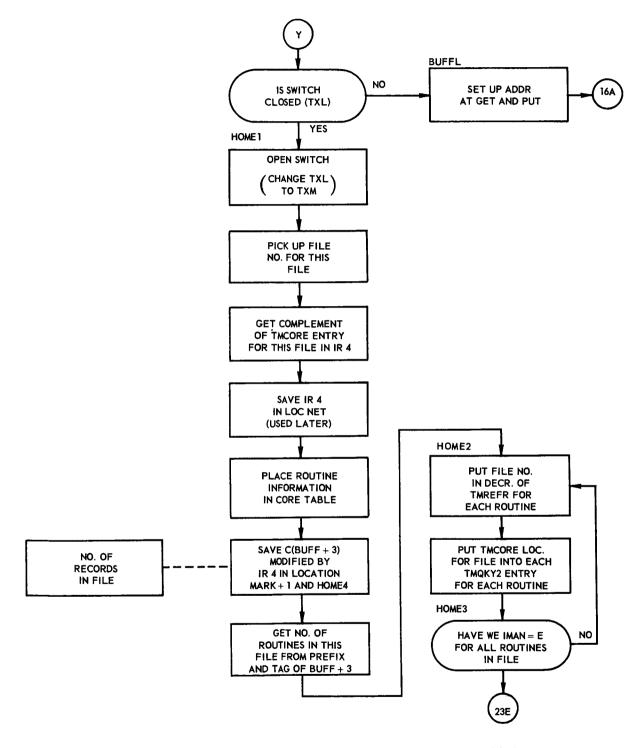


FIGURE 4-4. MXLOAD PROGRAM FLOW CHART (Sheet 5 of 14)

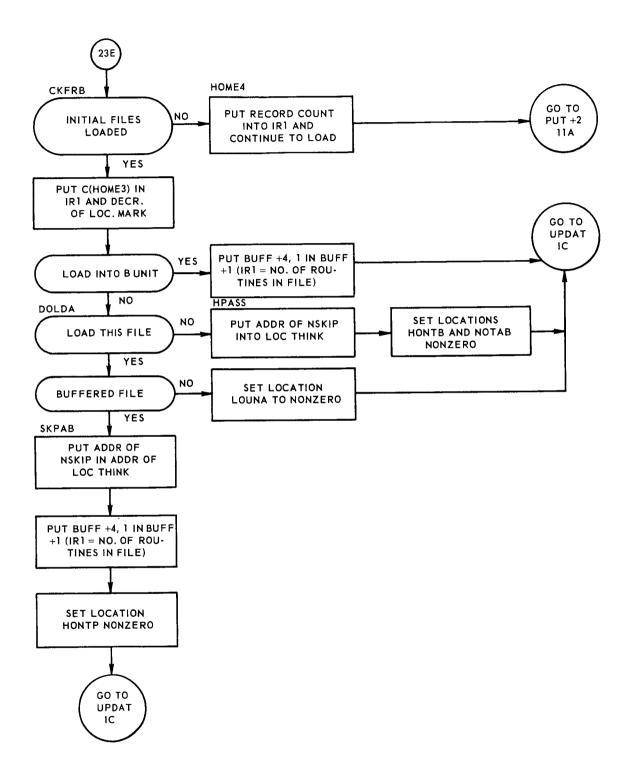


FIGURE 4-4. MXLOAD PROGRAM FLOW CHART (Sheet 6 of 14)

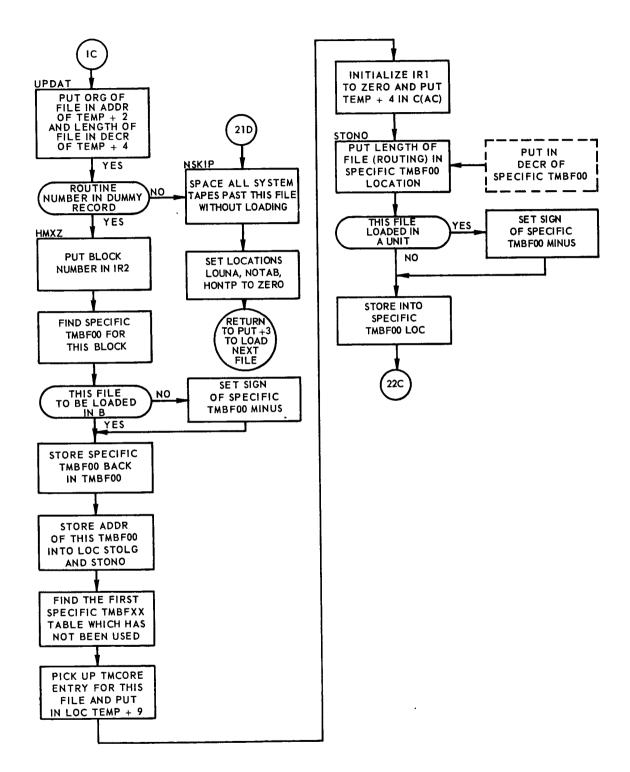


FIGURE 4-4. MXLOAD PROGRAM FLOW CHART (Sheet 7 of 14)

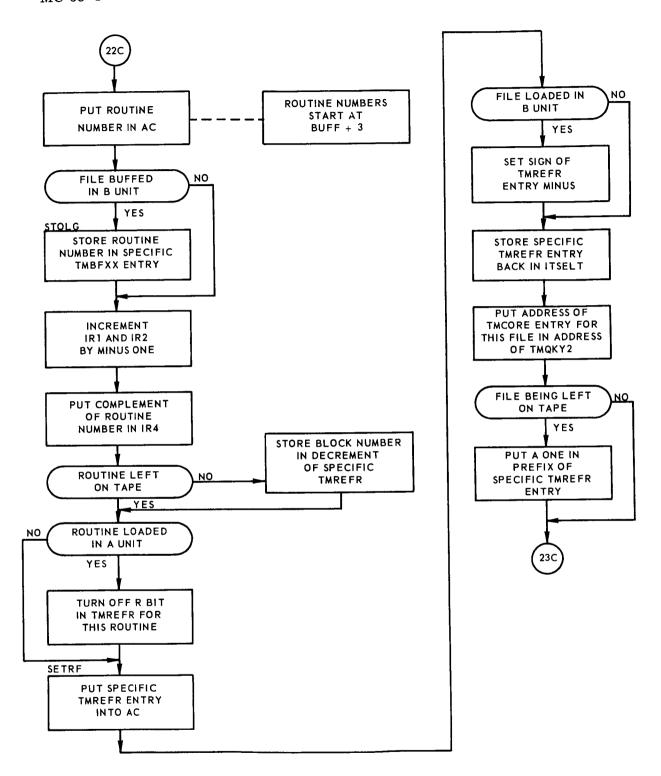


FIGURE 4-4. MXLOAD PROGRAM FLOW CHART (Sheet 8 of 14)

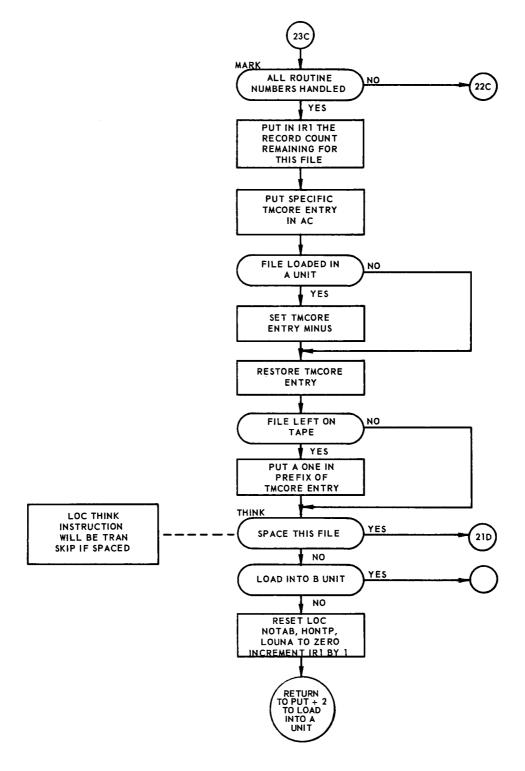


FIGURE 4-4. MXLOAD PROGRAM FLOW CHART (Sheet 9 of 14)

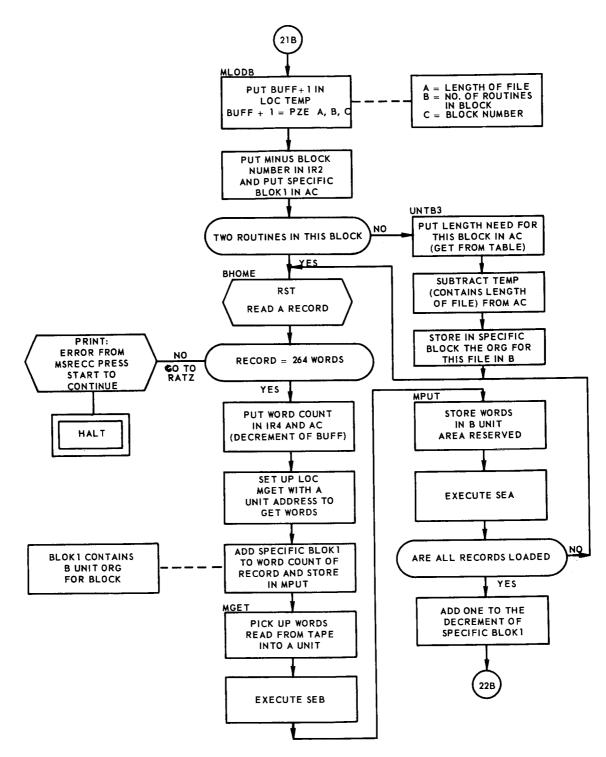


FIGURE 4-4. MXLOAD PROGRAM FLOW CHART (Sheet 10 of 14)

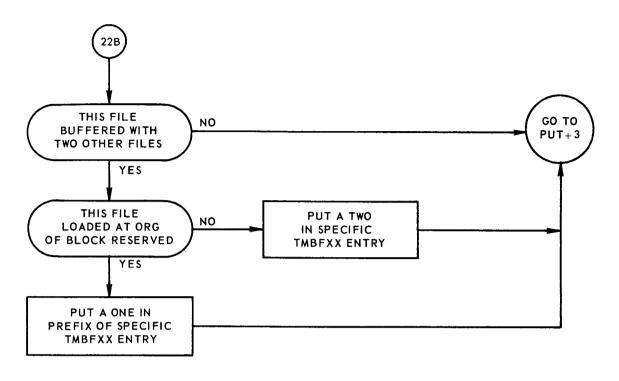


FIGURE 4-4. MXLOAD PROGRAM FLOW CHART (Sheet 11 of 14)

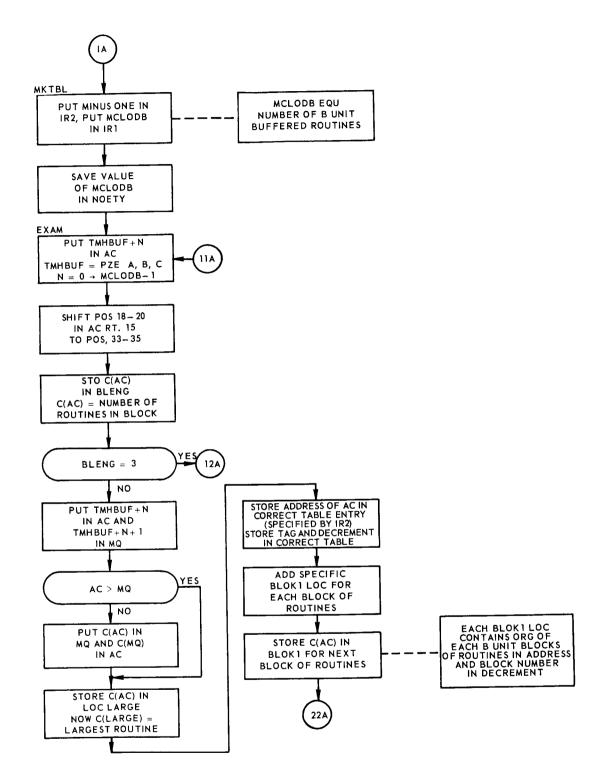


FIGURE 4.4. MXLOAD PROGRAM FLOW CHART (Sheet 12 of 14)

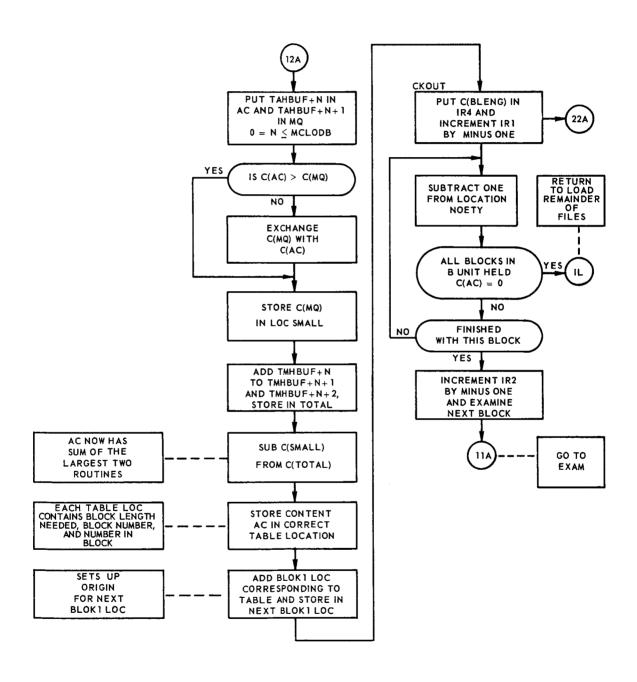


FIGURE 4-4. MXLOAD PROGRAM FLOW CHART (Sheet 13 of 14)

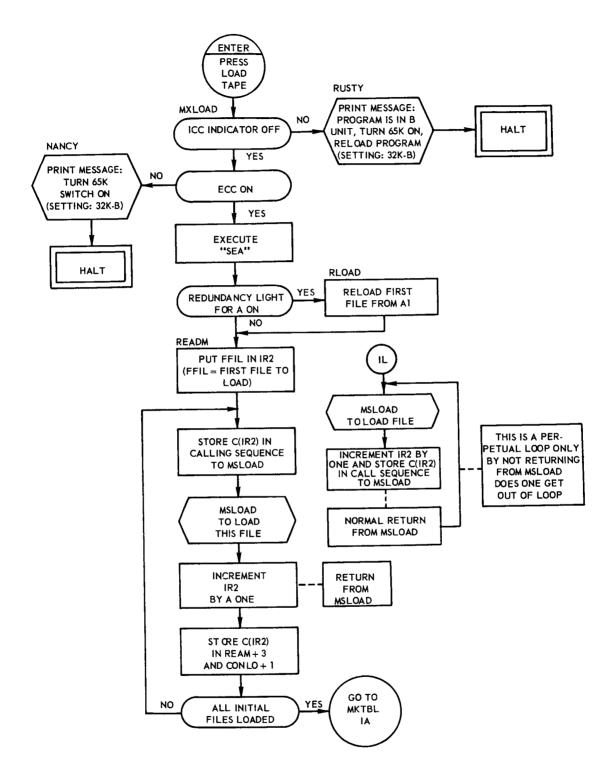


FIGURE 4-4. MXLOAD PROGRAM FLOW CHART (Sheet 14 of 14)

4.4 EXTENDED DEFINITION OF SYMBOLS PROCESSOR (MXDEFN)

MXDEFN automatically extends the definition of specified symbols among "n" separately compiled jobs during program execution. Thus, these symbol definitions are made available for referencing from any and/or all jobs.

Ths flow chart for MXDEFN is shown in Figure 4-5.

4.4.1 Input Requirements

MXDEFN requires as input at least one QDEFN macro. The QDEFN macro definition is as follows:

QDEFN	MACRO	ENTRY, ENO, XSYMB
	TCD	VCINT
	BCI	1, ENTRY
	FVE	ENTRY,, XSYMB
XSYMB	ORG	TMDEFN + ENO - 1
	END	

where the parameters are

ENTRY—the 6-letter symbolic name of a routine, processor, subroutine, etc., that requires an extended definition.

ENO—the sequence number or count.

XSYMB-the symbolic address of the extended definition of ENTRY.

A system example in the use of the QDEFN macro is:

ODEEN	MYHSOD,	1	TOPHYX
QDEFN	MINDOD.	ı.	AIDSOD

which expands into

	TCD	VCINT
	BCI	1, MYHSOD
	FVE	MYHSOD, , XYHSOD
XYHSOD	ORG	TMDEFN

4.4.2 Output Requirements

Output from MXDEFN consists of an extended definition table and several diagnostic messages.

a) TMDEFN—a table of extended definitions of the form TRA ENTRY. One core location for each QDEFN macro is reserved automatically.

b) Diagnostic Messages

- 1) QDEFN ENTRIES, NONDEFINED—each ENTRY of the QDEFN macros that is not defined in either "n" compiled jobs is listed following this heading.
- 2) QDEFN ENTRIES, MULTIDEFINED—each ENTRY of the QDEFN macro that is defined in two or more jobs is listed following this heading.
- 3) REPOSITION QDEFN ENTRIES SO THAT LAST QDEFN MACRO HAS GREATEST NUMERICAL ENO. DO NOT CONTINUE. If the last QDEFN macro does not have the greatest numerical ENO and if ENTRY of the QDEFN macro with the greatest numerical ENO is defined in either job, the extended definition for ENTRY is not determined and the message above is printed.
- 4) UNDEFINED SYMBOLS BETWEEN, AND INCLUDING, THE FOLLOWING TWO SYMBOLS ARE QDEFN ENTRIES. The first and last symbol processed by MXDEFN that appear in the diagnostic undefined listing of SOS are printed following this heading for each of "n" jobs. All QDEFN diagnostic messages are printed on-line and written on output tape A2 for off-line printing.
- 5) MXDEFN OVERLAPS TMDEFN TABLE. RELOCATE TMDEFN TABLE TO HIGHER CORE LOCATION. DO NOT CONTINUE. This heading is printed whenever overlapping occurs. Overlapping results because the TMDEFN table is located within a file that overlaps MXDEFN.

4.4.3 Method

The logic of extended definitions takes advantage of the method employed by SOS in assigning absolute values to undefined symbols within a job at compile time.

The address of the last word in a file plus one, or more precisely, the value of the location counter at the end of a file is assigned to the first undefined

symbol encountered within that file. This value is continuously incremented by one and assigned to each subsequent undefined symbol within the file.

By strategically placing ENTRY, ORG, TCD (TCD creates a file) in this order (see input requirements) the absolute value assigned to undefined symbols may be predetermined and controlled. Utilizing this logic SOS is forced to assign each undefined ENTRY of the QDEFN macro to a unique address within TMDEFN. A comparison routine, MXDEFN, processes all requested QDEFN ENTRIES per job, and their correct definitions, when found, are stored in their unique locations in TMDEFN.

The requested symbols for extended definitions normally are addresses of unconditional transfers. TMDEFN, which is saved and restored throughout the processing of "n" jobs, becomes a table of extended definitions during execution time of system runs.

4.4.4 Usage

Entry to MXDEFN is via the address, VCTSX, of the first TCD card in the system. Subsequent entries are to VCINT (the initialization section of MXDEFN) after reading each QDEFN record from SOS erase tapes. Exit is to MXSTW1, the system writer.

In addition to processing ENTRY symbols, MXDEFN determines the job of a multiple job system and rewinds ABS system tape A 6 if Job 1 is being processed.

- a) Storage Required-268 locations.
- b) MXDEFN Uses:
 - 1) Macro-QDEFN
 - 2) Subroutines:

External—SOS READ FILE, PRINT, and MSRECC Internal—VCWTD

- 3) Parameters-ENTRY, ENO, XSYMBL, and MNDEFN
- 4) Communication Cell-JBIND
- 5) Constant-VCTWO
- 6) Absolute Locations—loc 2, loc 3000₁₀, loc 3001₁₀
- 7) Tables:

External—TMDEFN
Internal—VCDFN

- 8) Mask-VCTRA
- 9) Internal Cells—VCRNT, VCFRT, and VCLST

c) Time Required (approximately):

Time
$$(\mu \text{ sec}) = \sum_{i=1}^{n} \sum_{j=1}^{m} [(T_1 + T_2) + K]$$

where

 $K = 156.96 \, \mu \text{ secs.}$ K = time required to initialize.

T1 = 87.2 μ secs. T₁ = time required to process one QDEFN RECORD if its ENTRY is undefined.

T2 = 106.8 μ secs. T₂ = time required to process one QDEFN RECORD if its ENTRY is defined.

M = variable M = number of QDEFN macros in one job.

N = variable N = number of jobs processed.

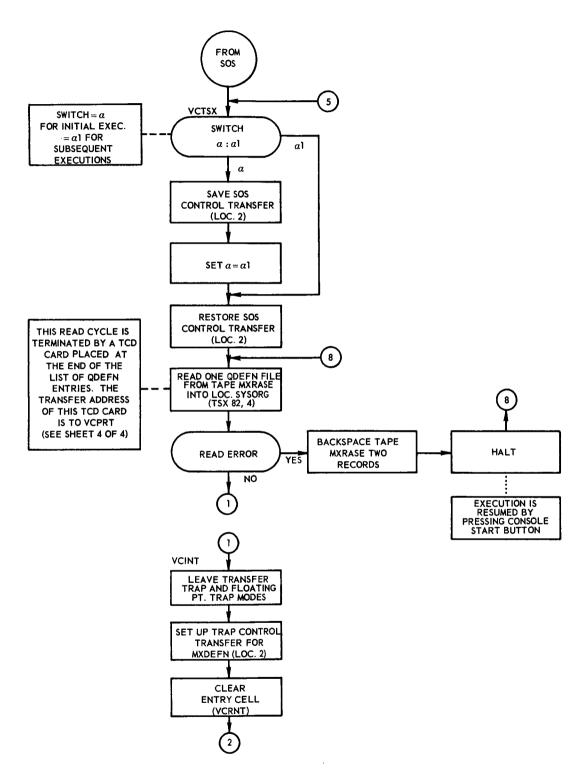


FIGURE 4-5. MXDEFN PROGRAM FLOW CHART (Sheet 1 of 4)

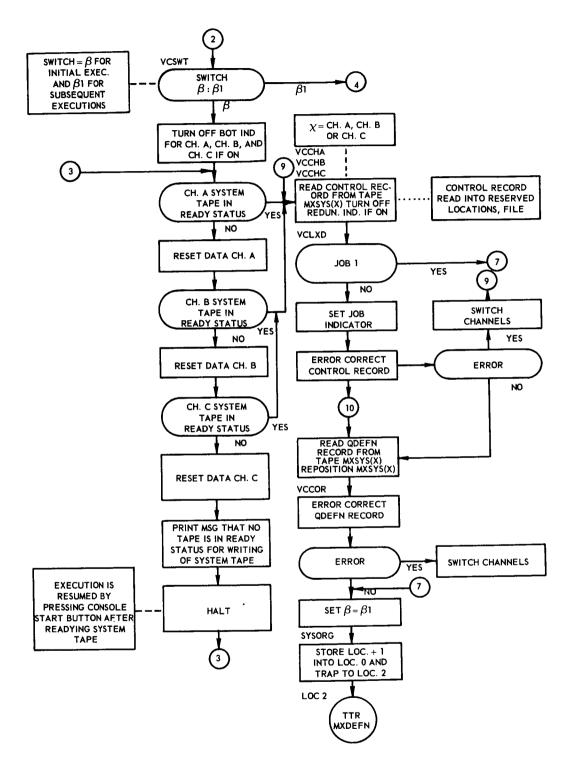


FIGURE 4-5. MXDEFN PROGRAM FLOW CHART (Sheet 2 of 4)

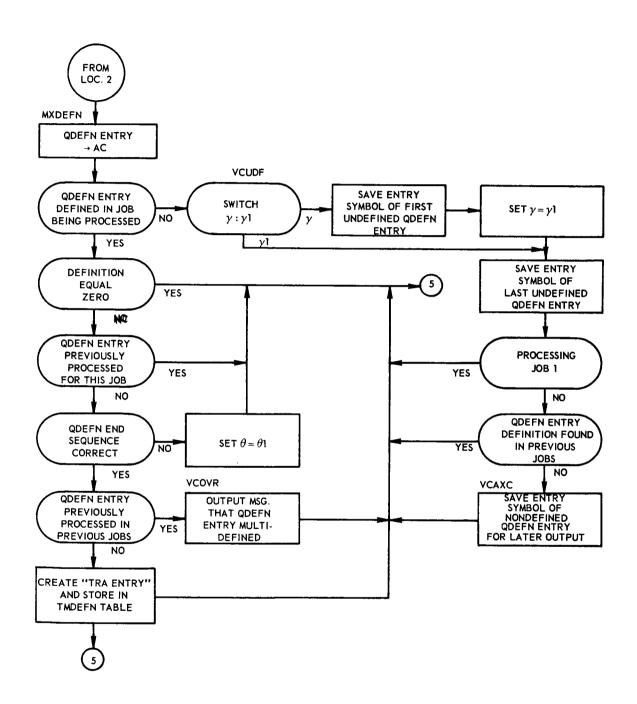


FIGURE 4-5. MXDEFN PROGRAM FLOW CHART (Sheet 3 of 4)

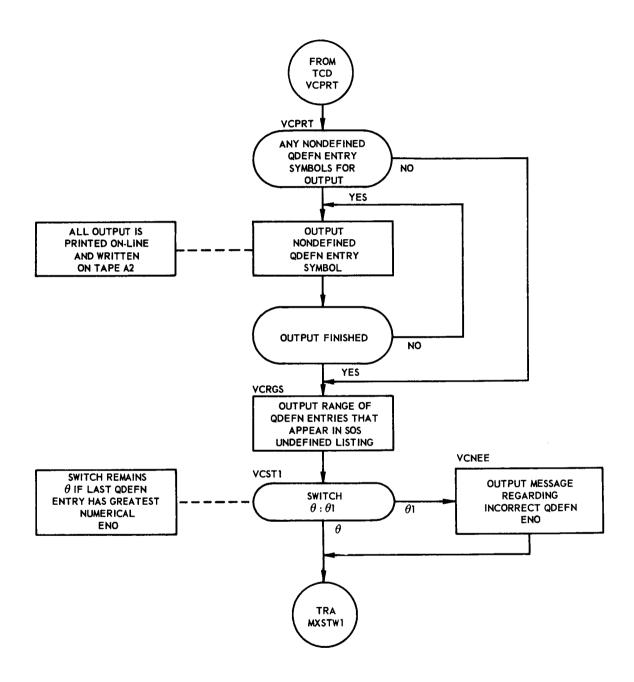


FIGURE 4-5. MXDEFN PROGRAM FLOW CHART (Sheet 4 of 4)

4.5 SYSTEM COMMUNICATION DURING DUAL COMPILATION (SETORG)

SETORG communicates common information between Jobs 1 and 2 during dual compilation and provides to the isolated tape writer program MXSTW1 the address and number of entries in BCOMTB, a table of definitions.

The flow chart for SETORG is shown in Figure 4-6.

4.5.1 Input Requirements

- a) Address of first word of the first record on the B4 tape contains the address of MXSTW1.
- b) Sense indicators are set with the mask 2304531, right-justified, if the run is a compilation.

4.5.2 Output Requirements

a) SYSORG—cell whose address is set:

alpha PZE ENDJB1

b) SYSFLO—table whose first word is set:

alpha PZE ENDJB1,, MOINIT

- c) MXSTW1-self-loaded into execution buffer.
- d) AC—address contains location of BCOMTB table and decrement contains table length.
- e) BCOMTB-table whose entries contain definitions for MXSTW1.

4.5.3 Method

SETORG receives control from SOS to establish common areas between Job 1 and 2 during dual compilations and to provide the entry to MXDEFN during program execution. When the run is a compilation, SETORG returns control to SOS after performing the necessary processing operations. With SOS load-and-go (LG) runs, however, MXDEFN is entered before control is returned to SOS. MXDEFN updates TMDEFN, a table of extended definitions.

When SETORG is entered during Job 2, MXSTW1 loads itself into core, receives control from SETORG, and processes entries in the BCOMTB table.

4.5.4 Usage

SETORG is entered from SOS and exits to either SOS or MXDEFN.

- a) Storage Required-60 locations.
- b) Time Required-0.082 milliseconds, excluding time to rewind B4 and to load MXSTW1.

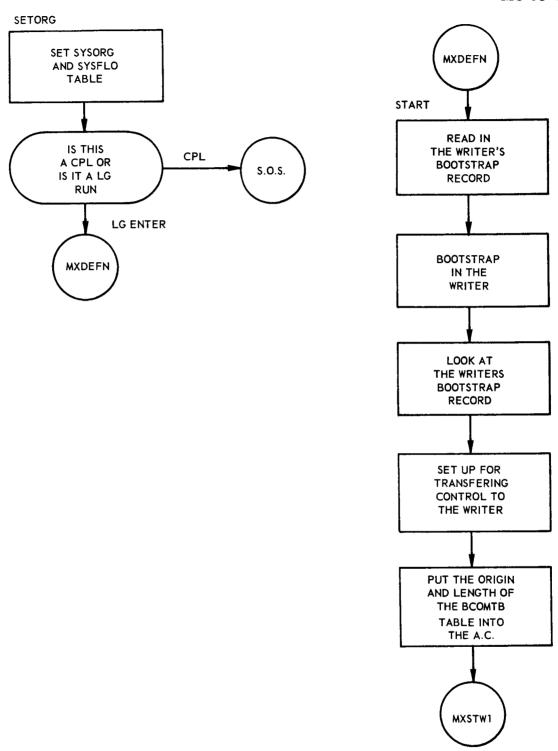


FIGURE 4-6. SETORG PROGRAM FLOW CHART

4.6 REAL TIME TRANSFER TRAPPING TEST PROGRAM (MTTEST)

MTTEST is used in the real time Mercury tracking system to locate the following types of program errors:

- a) If certain location(s) are being erroneously changed, MTTEST can determine the instructions that caused the error and provide for a dump to give the machine condition at the time of the error.
- b) If control is being erroneously transferred to a certain routine, a certain part of a routine, or to a certain location in core, MTTEST can determine the exact transfer instruction that caused the error and provide for a dump to give the exact machine condition at the time of the error.

Five other programs are included with MTTEST: MTTES4, MTTESA, MTTESB, and MTINIT—all are considered to be part of MTTEST.

The MTTEST program flow chart is shown in Figure 4-7.

4.6.1 Input Requirements

- a) MXCHNA—cell set for MTTEST by the channel A users. Each channel A processor, when it gains control, stores the location of its trap processor in the address of MXCHNA. MTTEST requires that MTTESA gain control whenever a channel A trap occurs. After MTTESA has been executed, it transfers to the correct trap processor by referencing MXCHNA.
- b) MXCHNB—cell set for MTTEST by the channel B users. This cell is the channel B counterpart of MXCHNA and is used by MTTESB.
- c) MXCHNC—cell set for MTTEST by the channel C users. This cell is the channel C counterpart of MXCHNA and is used by MTTESC.
- d) MCPROC—nonzero if the machine is inhibited; zero otherwise. Because a transfer trap can occur while the computer is inhibited, MTTEST must know the status of the computer so that it can return control with the computer in the proper mode. MCPROC is set to nonzero by MTTES4, MTTESA, MTTESB, and MTTESC and is reset to zero by MOPRIO.
- e) MCTRAP—nonzero if the computer is disabled. This cell is set and reset by the two Mercury system macros QENB2 and QENBA, respectively.

- f) Location 0—in the transfer trap mode, this location contains the address of the last transfer instruction that caused a trap.
- g) ZZZETM—location which is nonzero if the machine is in the transfer trap mode; zero otherwise. Because of timing, it may be desirable to have the computer in the transfer trap mode only for specific routines in the real time system. When this is the case, the two instructions STL ZZZETM and ETM must be removed from MTINIT. Then, each routine in the transfer trap mode must execute those two instructions on entry and the instructions STZ ZZZETM and LTM on exit. (Note, however, that if a routine being executed in the transfer trap mode is interrupted by other than a transfer trap, the entire Mercury system will operate in the transfer trap mode until control is returned to the interrupted routine and its execution is completed.)

4.6.2 Output Requirements

- a) Location 3—contains trap information when a trap occurs on the DCC or another external device. If MTTES4 detects that control was trapped from location 1, MTTEST, MTTEST + 1, or MTTEST + 2 (before MTTEST could disable the machine) MTTES4 computes the effective address in the Mercury system to which MTTEST would have returned, and stores this address in location 3. After performing all of MTTEST's duties, MTTES4 transfers to the external trap control routine MORTCC. With this method, the logic proceeds as if MTTEST had completed its work and the external trap to location 4 had not occurred until MTTEST had returned control to the Mercury system. All of this is necessary because once a transfer trap occurs, MTTEST must complete all of its functions before allowing another transfer trap.
- b) Location 11—contains trap information when a trap occurs on channel A. MTTESA gains control from a trap on channel A and acts on location 11 in a way completely analogous to the way MTTES4 processes location 3.
- c) Location 13—channel B trap location cell. This location receives the output of MTTESB (see parts a) and b)).
- d) Location 15—channel C trap information cell. This location receives the output of MTTESC (see parts a) and b)).

Other Programs used by MTTEST

- a) MTTES4—routine which processes all external traps to location 4. MTTES4 checks to see if an external trap took control from MTTEST (before MTTEST had time to disable the machine). If control was interrupted from location 1, MTTEST, MTTEST + 1, MTTEST + 2 MTTES4 performs all of the duties of MTTEST and stores in location 3 the address in the Mercury system to which MTTEST would have returned had it not been interrupted. Control is then given to MORTCC, the Mercury external trap processor. In this way, every transfer trap is processed before control is returned to the Mercury tracking system.
- b) MTTESA—routine which processes all channel A traps. MTTESA checks if a channel A trap took control from MTTEST (before MTTEST had time to disable the computer). After performing the same operation as MTTES4, MTTESA transfers to the correct trap processor specified by the cell MXCHNA.
- c) MTTESB—routine which processes all channel B traps. MTTESB processes channel B in a way entirely similar to the way MTTESA processes channel A.
- d) MTTESC-routine which processes all channel C traps.
- e) MTINIT—initializing routine. MTINIT receives control from the real time initialization program M0INIT and sets up all trap control locations with the correct trap transfers. It dynamically changes all those programs which are in core initially and which reference the trap control locations 1, 4, 11, 13, or 15. MTINIT then sets ZZZETM to nonzero to indicate the transfer trap mode, enters the transfer trap mode, and trap transfers to M0PRIO.

4.6.3 Method

As long as the Mercury system is in the transfer trap mode, MTTEST receives control (by way of a trap) every time a transfer instruction is executed. In the case of a conditional transfer instruction (TIX, TNX, TXH, TXL, TOV, TNO, TPL, etc.), the transfer trap occurs only if the transfer condition is met.

Every time MTTEST receives control it checks the error condition under investigation. If the error condition is present, the computer halts (HTR*). At this point a dump should be taken. Location ZZANS in MTTEST will contain the location of the transfer instruction at which the error was detected, and ZZLAS will contain the location of the transfer instruction executed previously.

The error condition is also checked every time an external trap or a channel trap occurs. Since MTTEST receives control on every trap (its routines MTTES4, MTTESA, MTTESB, and MTTESC receive control on external and channel traps) it can always finish processing a transfer trap before another transfer trap occurs. With this method, the error condition can be checked every 4 or 5 instructions (assuming this is the average span between transfer instructions) throughout the entire real time system.

There are, however, restrictions to this method of debugging. First, using MTTEST changes the time requirements of the real-time programs by, at times, a factor of 10. In the orbit and reentry phases this causes no problem since there is spare time. In the launch phase, however, the extra time needed to run the entire system in the transfer trap mode should be limited to a few "suspected" routines. (Refer to the description of the cell ZZZETM under Input Requirements for the method of effecting this.)

The other restriction to the use of MTTEST also lies with timing. If the nature of the error is such that it only occurs when certain routines are executed in a precise time sequence, the error might not occur when the machine is in the transfer trap mode and the time sequence has changed.

4.6.5 Usage

To use MTTEST, the programmer must write the instructions for the particular test that he wishes to perform. IR 2, the AC, the OV/UN indicator, and the Sense indicators are saved by MTTEST. All other registers and indicators used by the programmer's test instructions must be saved and restored by the test program. The test program must follow the MTTEST deck.

If no error condition is found, the program should transfer to "1, 4" to return to normal program flow. If an error is found, the test program should transfer to "ERROR". Note: Locations 4, 11, 13, and 15 are changed by MTTEST so that:

Location 4 contains	TTR	MTTES4
Location 11 contains	TTR	MTTESA
Location 13 contains	TTR	MTTESB
Location 15 contains	TTR	MTTESC

Example 1: To determine which instructions change location 4.

MTTEST

DECK

	CLA	4
	CAS	TEST
	TRA	ERROR
	TRA	1, 4
	TRA	ERROR
TEST	TTR	MTTES4

When a change in location 4 is detected, the program will come to a halt (HTR*). When a dump is taken, location ZZANS in MTTEST will contain the location of the transfer instruction at which the error was detected and ZZLAS will contain the location of the transfer instruction that had been executed previously. Therefore, the instructions that caused the error will lie between the location to which the transfer instruction (designated by location ZZLAS) was transferring and the location in ZZANS. With this method the error can usually be narrowed to 4 or 5 instructions, and the routine causing the error will always be detected since control cannot leave a routine except with a transfer instruction.

Example 2: To determine which transfer instruction transfers to the routine M0ENDS. M0ENDS is followed by the routine M0PANL.

MTTEST

DECK

	SXA	A, 4
	TSX	ZZABS, 4
	TXL	A, 2, M0ENDS-1
	TXL	ERROR, 2, M0PANL-1
A	AXT	**, 4
	TRA	1, 4

ZZABS is a subroutine which computes the absolute (effective) address to which the transfer instruction (which caused the trap) is transferring. The subroutine takes into account indirect addressing and index modification and stores the absolute address in IR 2.

When the program finds a transfer instruction that transfers to or into M0ENDS, it will halt disabled (HTR*). When a dump is taken, location ZZANS

in MTTEST will contain the location of the transfer instruction transferring to ${\tt M0ENDS}.$

To execute MTTEST, the MTTEST program and the test instructions should be placed in the Mod deck for MXMRGE and everything else should proceed as in a normal run. When the program finds the error condition present it will halt disabled (HTR*). At this point a dump should be taken.

Since MTTEST is altered into the compilation, care must be taken that the MTTEST deck corresponding to the correct compilation is used. If a new compilation is being used, the existing alter numbers in the MTTEST deck must be updated to correspond with the new compilation. Wherever possible, CHANGE cards have been used to reduce the updating process.

- a) Storage Required—201 locations. Thirty-six of these locations (the MTINIT program) are erasable immediately after initialization of the Mercury real time system.
- b) Time Required:
 - 1) MTTEST: maximum—.168 millisecond minumum—.139 millisecond
 - 2) MTTES4, MTTESA, MTTESB, or MTTESC (all require the same amount of time): maximum—.241 millisecond minimum—.213 millisecond
 - 3) MTINIT (Only executed once at initialization):

.081 millisecond

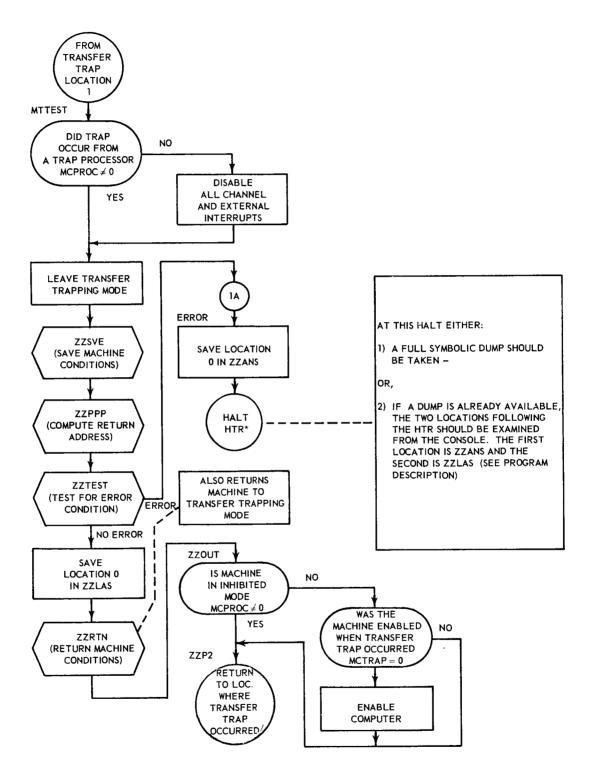


FIGURE 4-7. MTTEST PROGRAM FLOW CHART (Sheet 1 of 9)

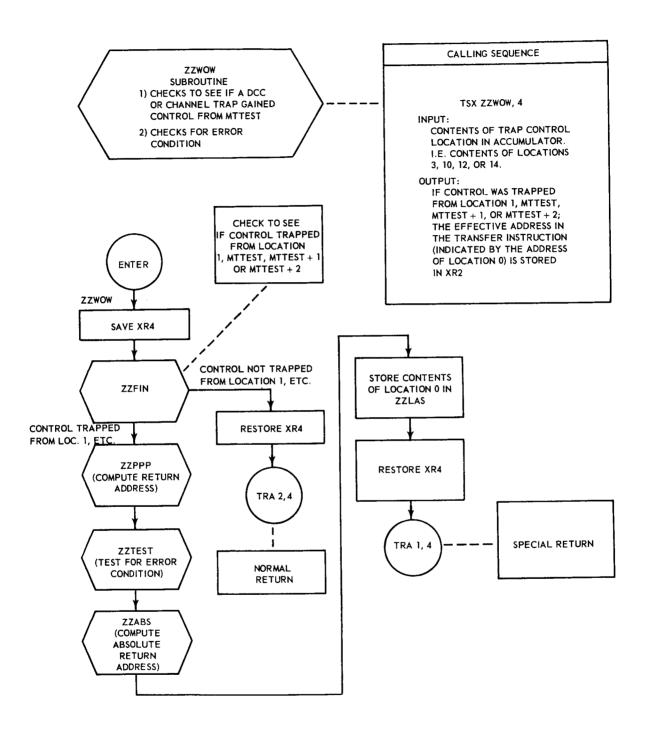


FIGURE 4-7. MTTEST PROGRAM FLOW CHART (Sheet 2 of 9)

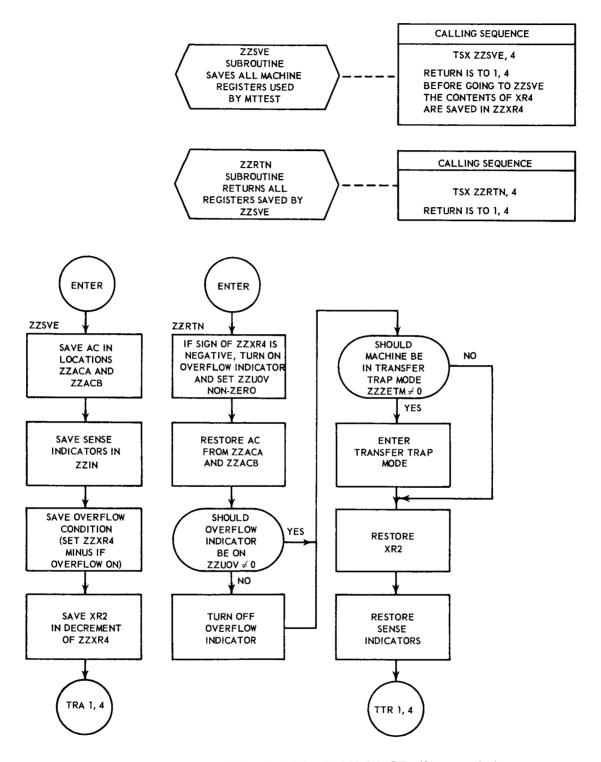


FIGURE 4-7. MTTEST PROGRAM FLOW CHART (Sheet 3 of 9)

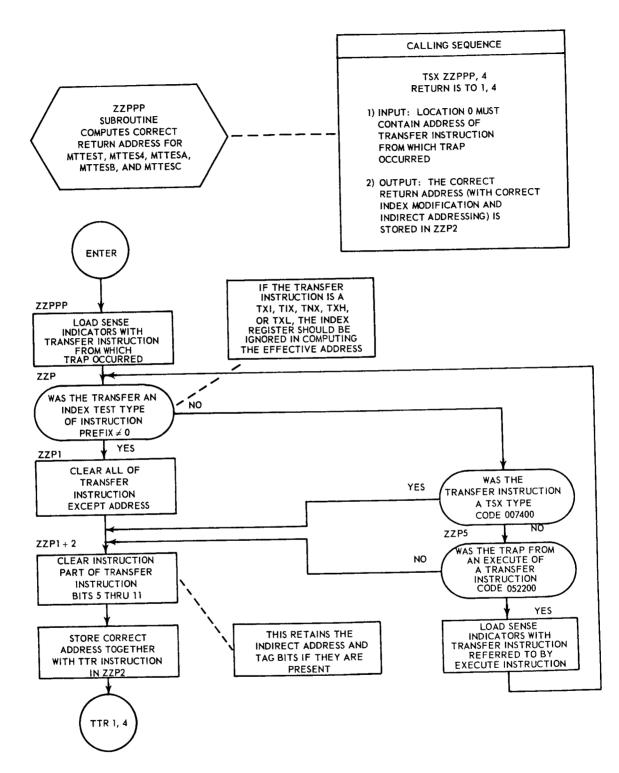


FIGURE 4-7. MTTEST PROGRAM FLOW CHART (Sheet 4 of 9)

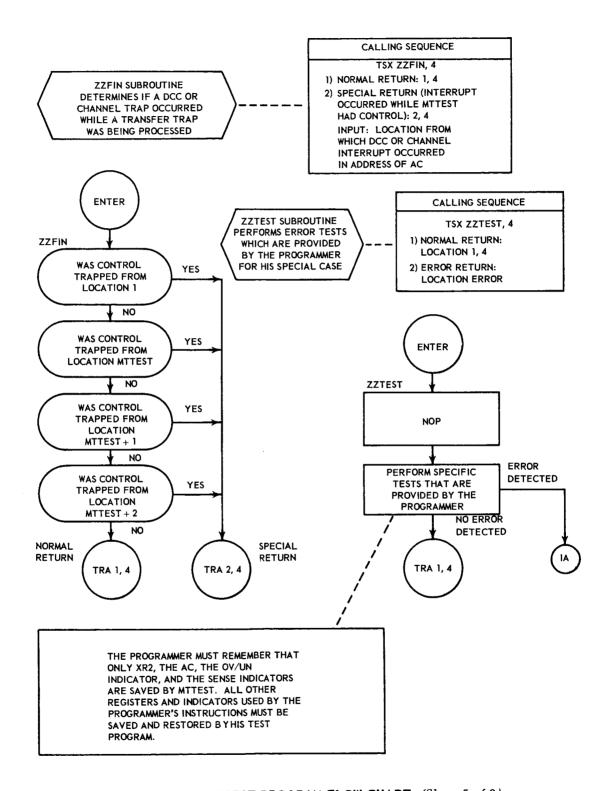


FIGURE 4-7. MTTEST PROGRAM FLOW CHART (Sheet 5 of 9)

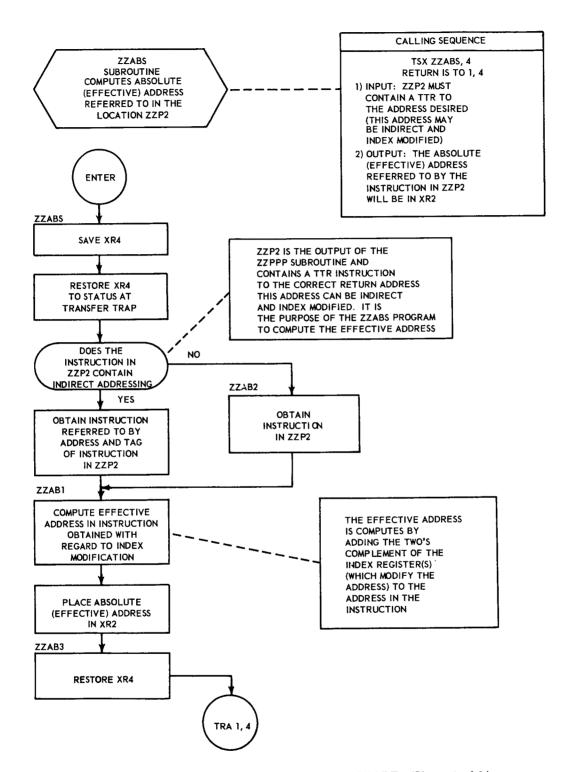


FIGURE 4-7. MTTEST PROGRAM FLOW CHART (Sheet 6 of 9)

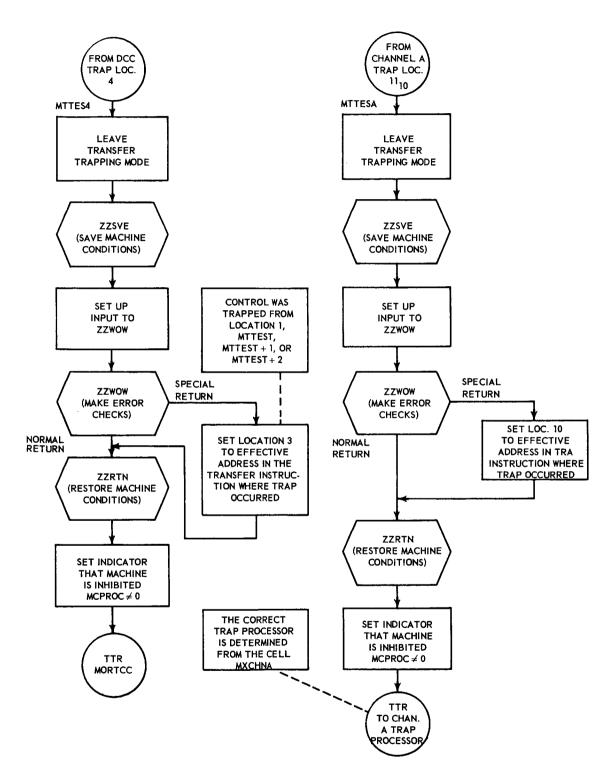


FIGURE 4-7. MTTEST PROGRAM FLOW CHART (Sheet 7 of 9)

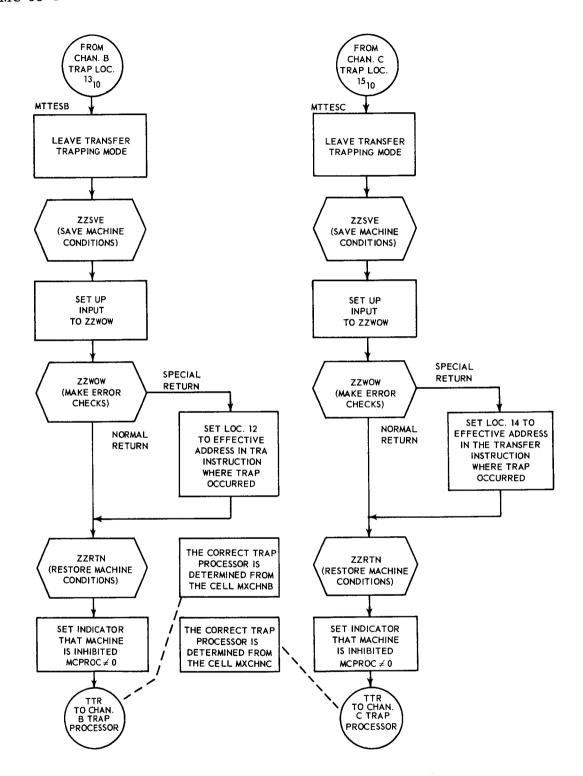


FIGURE 4-7. MTTEST PROGRAM FLOW CHART (Sheet 8 of 9)

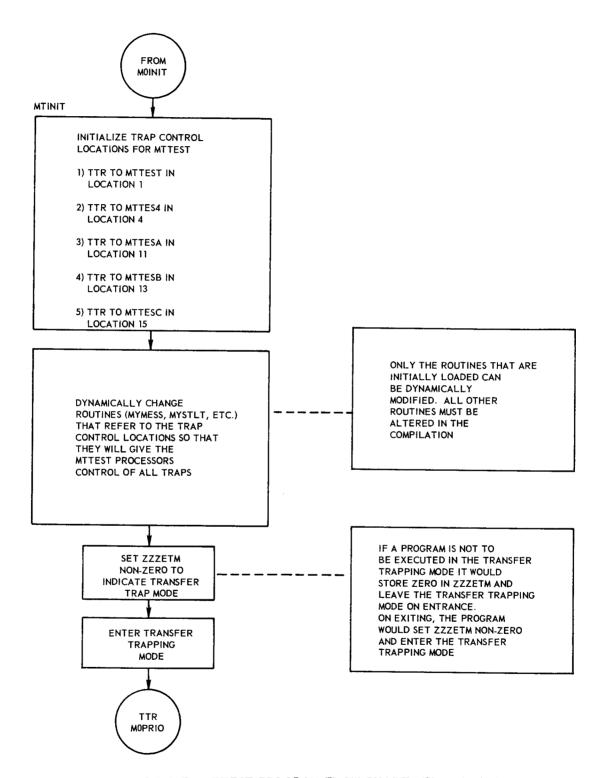


FIGURE 4-7. MTTEST PROGRAM FLOW CHART (Sheet 9 of 9)

4.7 PROGRAM TO WRITE THE ISOLATED WRITER PORTION OF THE B4 TAPE (WRTB4T)

WRTB4T writes, and permits modifications to, the first three files on the B4 tape which contain programs used to write the operational Mercury system tape.

The flow chart for WRTB4T is not shown because of its simplicity.

4.7.1 Input Requirements

Input to WRTB4T is the complete MXSTW1 program.

4.7.2 Output Requirements

Output from WRTB4T is the first three files of the B4 tape. The first file of two records—one has the self-loading instructions for MXSTW1, the other has the MXSTW1 program; the second file contains the single-record MXLOAD program; and the third file consists of error-correcting and printing programs.

4.7.3 Method

WRTB4T is entered by SOS with the loading subroutine HBLEW after MXSTW1 has been loaded into core. WRTB4T writes the files on B4 and halts with 77776_8 in the AC.

4.7.4 Usage

WRTB4T is entered from HBLEW and halts with 77776_8 in the AC.

Storage required—27 locations.

4.8 PROGRAM TO WRITE DUMPING PORTION OF B4 TAPE (HOMER)

HOMER writes, and permits modification to, the dumping programs on the B4 tape used during the execution of the Mercury operational system.

The flow chart for HOMER is not shown because of its simplicity.

4.8.1 Input Requirements

Input to HOMER includes an SOS SYSMIT tape produced from a dual compilation dumping deck. Also, the first three files of the B4 tape must have been written by WRTB4T.

4.8.2 Output Requirements

Output from HOMER consists of the following files on the B4 tape:

- a) Fourth file—SOS IBMONITOR and SNAP
- b) Fifth file—SNAPOR
- c) Sixth file-DUAL DUMP CORING and SNAP, origined at 3000
- d) Seventh file-DUAL DUMP CORING and SNAP, origined at 30

4.8.3 Method

HOMER and WRTB4T write the entire B4 tape. HOMER is part of the B1 tape and is entered from SOS. When necessary, HOMER rewinds the B4 tape, then spaces the first three files before writing the dumping programs. The dumping programs are read from the B1 SYSMIT tape and are rewritten on the B4 tape in executable format.

4.8.4 Usage

HOMER is entered from and exits to SOS.

a) Storage Required—111 locations

- b) Erasable Locations-
 - 1) SOS IBMONITOR and SNAP-0 to 5672_8
 - 2) SNAPOR -13225_8 to 13423_8
 - 3) DUAL DUMP CORING and SNAP-5670 $_8$ to 11476 $_8$
 - 4) DUAL DUMP CORING and SNAP-36 $_8$ to 1413 $_8$
- c) Time Required—depends on reading and writing times

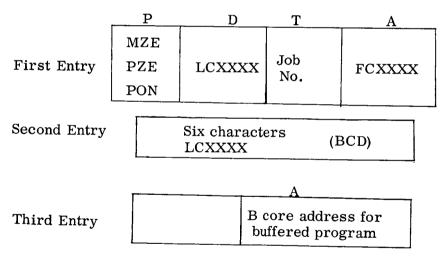
4.9 ISOLATED DUMPING PORTION OF B4 TAPE (ISODMP)

ISODMP, files 4 through 7 of the B4 tape, consists of a number of separate programs used to produce a symbolic dump of the Mercury operational system.

The flow chart for ISODMP is shown in Figure 4-8.

4.9.1 Input Requirements

- a) A file of dictionaries for N jobs must be contained on the Mercury system tape.
- b) AC-contains location of TMCORE.
- c) MQ—contains location of TMENDS (a table containing machine conditions at time of request for dump).
- d) TMCORE—a variable length table listing the location and status of all programs in storage. This table is built and maintained by MXLOAD, MYBUFR, and MYRSYS; and each file on the system tape is defined in in the table with three entries:



where the first entry is defined:

MZE-file in A core

PZE-file in B core

PON-file on tape

LCXXXX—last location + 1 of file

Job No. —compilation (job) No.

FCXXXX-first location (origin) of file

4.9.2 Output Requirements

Final output from ISODMP is produced on A2.

4.9.3 Method

The ISODMP programs are loaded by SGENDX, which first writes all of core on B5 and then loads the seventh file of the B4 tape (the DUALDP and CORIN programs) beginning at location 00036. DUALDP is initialized and, in turn, it modifies the SOS program SNAP so as to establish entries to dumping programs required for subsequent operations. After modifying SNAP, DUALDP proceeds to dump the A and B cores of the 65K unit.

NOTE

Files 6 and 7 of the B4 tape contain nearly identical DUALDP and CORIN programs; in addition, file 6 has the SOS SNAP program. The file 6 SNAP program, however, is read into a different area from that used by the SOS version, and permits dumping of core below location 3000. File 7 is used with the SOS SNAP program contained in file 4.

The symbolic dump is basically the same for dual and multi-compiled systems; however, some differences exist and are described in the following:

Dual Compilation

The DUALDP program, in file 7, writes snaps of Job 1 on B2 and of Job 2 on B10. The program then writes the dictionaries for Job 1 and Job 2 on tapes B1 and B3, respectively, and halts. The SNAPTRAN program from SOS is entered, correlates the B1 and B2 tapes with their dictionaries, and records the results on A2.

Multi-Compilation

DUALDP, in file 7, dumps selected areas of storage on B2, reads the dictionaries from the Mercury system tape, and then halts. SNAPTRAN is entered, correlates the information, and records the results on A2.

In both dual and multi-compiled systems, DUALDP first dumps areas from 3000 to 32767. After these areas have been dumped, control is given to BSNAP to dump B core. BSNAP is a subroutine in the CORIN program and dumps the B-core portion of the Mercury system. When BSNAP has finished processing B core, control is returned to DUALDP in file 7. DUALDP reads in file 6, starting at location 3000, and gives control to the DUALDP program contained in file 6. This version of DUALDP loads the first 3000 words from the B5 tape

back into storage, starting with location 0. (This data was replaced by programs needed for dumping and is now returned to storage to be processed.) This area (0-3000) is then cored and control is returned to DUALDP which reads in file 4 of the B4 (containing SOS IB MONITOR and SNAP). Control is then transferred to programs in file 4.

4.9.4 Usage

ISODMP is initialized manually at the computer console:

- a) To obtain a final Mercury system dump, set SS 6 down and depress Entry key 2 or manually transfer to 77777.
- b) Blank tapes should be mounted on A2, B1, B2, and B5 for multi-compiled systems; and on A2, B1, B2, B3, B10, and B5 for dual compiled systems.
- c) The standard SOS system tape must be present on channel A, and the standard B4 tape must be dialed on B4.
- d) After dumping has been completed, the program halts. For both dual and multi-compiled systems, the standard SOS system tape must be dialed to A1; for multi-compiled systems only, the absolute system tape must also be dialed to A10.
- e) The final stop of a successful dump is 00173—other halts should be accompanied by an on-line error message from SOS.

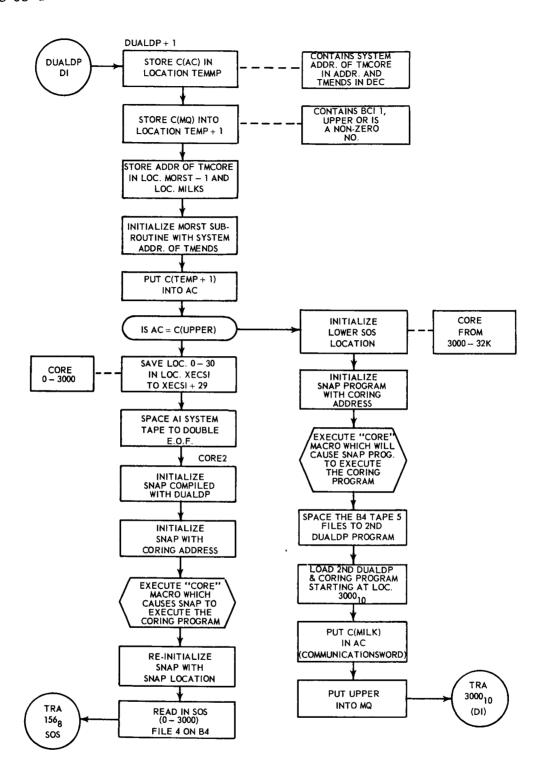


FIGURE 4-8. ISODMP PROGRAM FLOW CHART (Sheet I of 2)

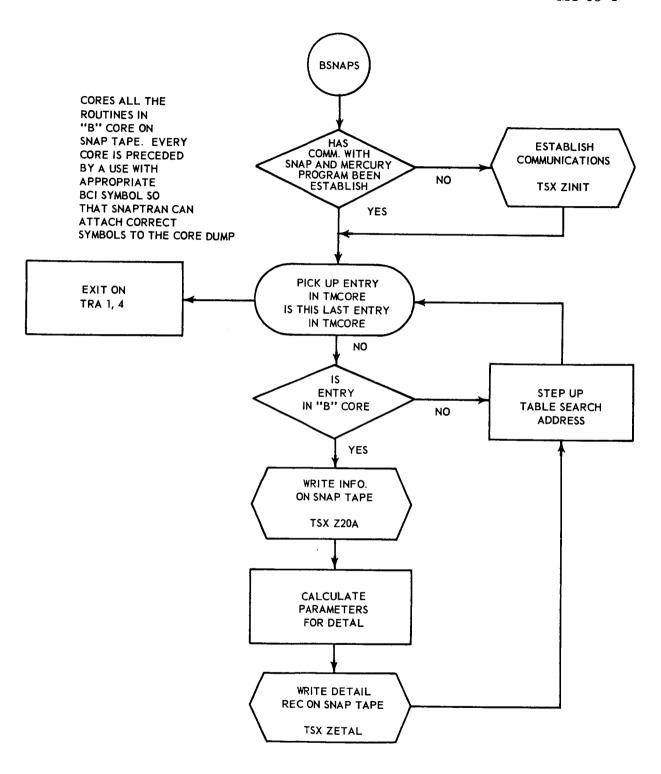


FIGURE 4-8. ISODMP PROGRAM FLOW CHART (Sheet 2 of 2)

4.10 DUMP PROGRAM READER (SGENDX)

SGENDX reads the dump program, DUALDP, and the SOS snap file from the B4 utility tape in preparation for a system dump.

The flow chart for SGENDX is shown in Figure 4-9.

4.10.1 Input Requirements

- a) TMCORE—table containing information for the dump program. SGENDX gives the location of this table to the dump program (the dump processor cannot reference TMCORE symbolically since the dump program is not included in the system compilation).
- b) TMENDS—table containing the status of the computer registers at the time of termination of the run. The registers are saved by M0PANL, the location of TMENDS is passed on to the dump program by SGENDX, and TMENDS is used by the dump program so that the computer registers appear in the dump.
- c) The B4 tape is read by SGENDX to obtain the SOS snap program (file 4) and the dump program (file 7).

4.10.2 Output Requirements

SGENDX writes the lower core radar blocks on B5 and gives the locations of TMCORE and TMENDS to the dump program.

4.10.3 Method

SGENDX receives control from M0ENDS and immediately disables the computer and deactivates the subchannels of the DCC. The lower core radar blocks are written on B5 (no longer necessary since M0ENDS writes all of lower core on B5 previous to giving control to SGENDX) and the B4 tape is searched for the SOS file. When the SOS file is found (the fourth file on the B4 tape), it is read into lower core. The dump program is then read into lower core below the SOS program (the dump program is the seventh file on the B4 tape). The AC is loaded with the addresses of TMCORE and TMENDS, the MQ is set to non-zero to indicate a real time run, and control is given to the dump program at location 30.

4.10.4 Usage

SGENDX is entered from M0ENDS and exits to DUALDP.

- a) Storage Required-48 locations
- b) SGENDX Uses:
 - 1) Tables—TMCORE and TMENDS
 - 2) Constant-K00000
 - 3) Tapes-B4 and B5
- c) Time Required-0.096 milliseconds (does not include tape transmission time)

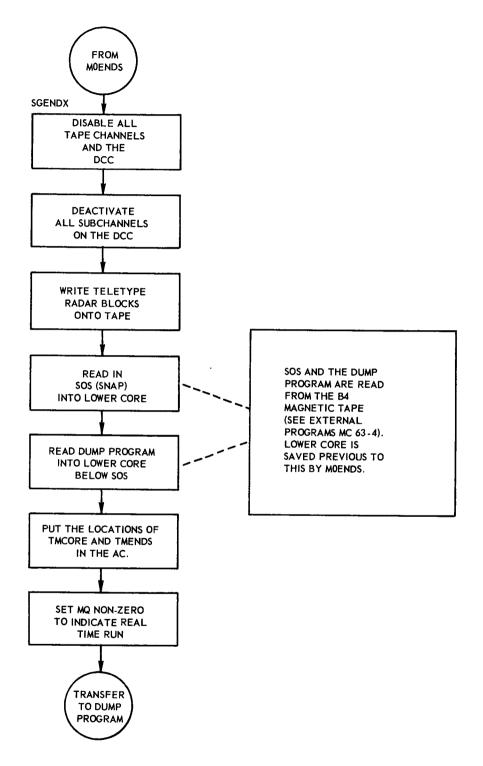


FIGURE 4-9. SGENDX PROGRAM FLOW CHART

4.11 PROGRAM TO INITIATE TAKING SNAP DUMPS OF THE MERCURY SYSTEM (CORING)

CORING is contained on the B4 tape and initializes the SOS SNAP program so that files from different compilations may be snapped on different tapes.

The flow chart for CORING is shown in Figure 4-10.

4.11.1 Input Requirements

Input to CORING includes the location of TMCORE, a variable length table written by MXLOAD and containing:

- a) The A core address of the first and last locations to be dumped.
- b) The BCD symbol of the last location + 1 of the file containing the area to be cored.
- c) The B core address of buffered programs.

4.11.2 Output Requirements

Output from CORING includes the USE and CORE macros and a means of returning control to the main program.

4.11.3 Method

When the CORE macro is entered, control is transferred to CORING. Using the parameters of the CORE macro, CORING performs a table look-up of TMCORE to determine which files lie between the core parameters and the compilations in which these files were compiled. Then CORING converts, if necessary, the original CORE macro into macros for the individual files contained within the limits of the original CORE macro. Each of the subcore macros is preceded by a USE macro with the BCD symbol of the last location of the file. The necessary instructions to initialize SNAP when successive files are from different compilations are also provided by CORING. All these instructions are assembled in a logical fashion, forming a small routine which transfers control to SOS to perform the actual snaps. The final instruction of this subprogram returns control to the original program initiating the CORE macro.

4.11.4 Usage

CORING is entered from the CORE macro.

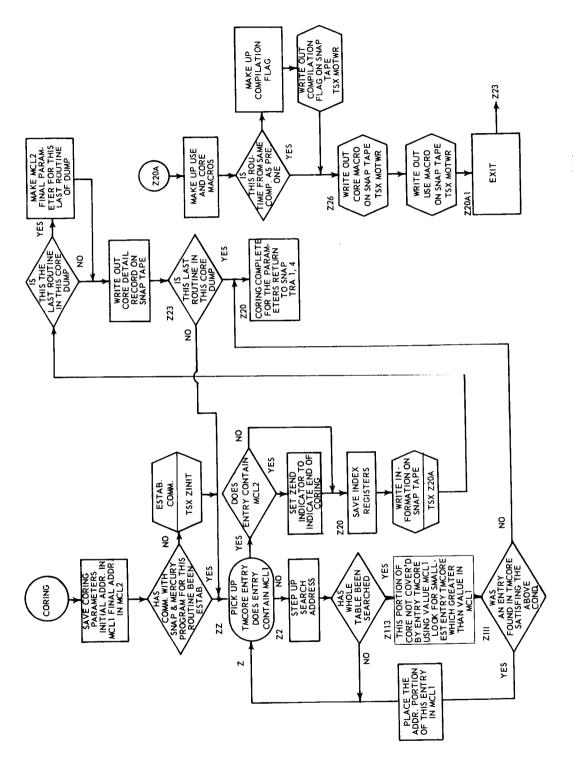


FIGURE 4-10. CORING PROGRAM FLOW CHART (Sheet I of 2)



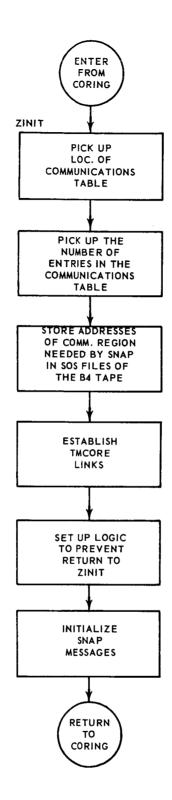


FIGURE 4-10. CORING PROGRAM FLOW CHART (Sheet 2 of 2)

4.12 MESSAGE TAPE WRITER PROGRAM (MXWMOT)

MXWMOT prepares the message tape for the Mercury Programming System.

The flow chart for the MXWMOT program is shown in Figure 4-11.

4.12.1 Input Requirements

Input to MXWMOT consists of a deck of data cards, each card containing one message. The cards must be numbered serially from 1 to 9999 in columns 3-6, right-justified. The symbolic (Hollerith coded) message must be placed in columns 11-72. Columns 1 and 2 must be blank or must contain zeroes.

4.12.2 Output Requirements

Each message is written as one record on the message tape. The record consists of 25 words: the first word contains the message number (columns 3-6) and the remaining 24 words contain the card image of the message (columns 11-72).

Input cards are listed when the execution program, MXWMOT, contains no modifications—MXWMOT is a standard SOS modify-and-load job in column binary squoze and may contain modifications. With modifications to MXWMOT there is no listing.

The card image is printed for any input message card which is not in numerical sequence. If any card is out of sequence, the message tape is invalid and the program must be rerun.

4.12.3 Method

During the mission the message tape supplies prepared messages on request from the tracking program. These messages signal significant events in the mission or mission computations and are printed on-line continuously during mission operations.

Table 4-1 contains the messages on the message tape, their associated number, and the processor in which the queue of each message occurs. Most of the messages are self-explanatory; however, there is an explanation of the ones that may be doubtful following the message in the list. A detailed discussion of the messages is included in the Goddard Monitor Programs manual, MC 63-2.

The message (columns 11-72) is read from the card and stored in the last 24 cells of a 25-cell block in core storage. The message number (columns 3-6) is stored in the first cell of the 25-cell block, and the complete block is written on tape A6 as one record.

4.12.4 Usage

- a) Operator Procedures:
 - 1) Ready the following tapes:

Tape	For MXWMOT	After MXWMOT
A1	SOS tape	SOS tape
A2	Blank tape	SOS output tape
A3	MXWMOT job tape	MXWMOT job tape
A6	Blank tape	Binary output message tape
B1	Blank tape	Pool tape
$\mathbf{B2}$	Blank tape	Pool tape
	1	

- 2) Ready the card reader with card messages (in numerical sequence).
- 3) Ready the on-line printer. Press CLEAR and LOAD TAPE buttons.
- 4) The program stops at 1402_8 . Remove A6 and label.
- b) Error Conditions—if the messages are not in numerical order, those out of order are printed on-line. If any messages are out of order, the message tape is invalid and the program should be rerun.
- c) Checkout—in testing the program, the squoze deck without any modification cards (in which case a printout of all the data cards was given) was run with three different sets of data. The first data set consisted of messages numbered 1-12 and a PAUSE card; the second data set consisted of 1, 2, 811, 4, 5, 1000, 7, 8, 8531, 10, 9999 and 12; and the third data set, of messages 1-12.

The squoze deck with modifications was run to delete the listing of all data cards; the program was run with two sets of data cards. The first set of data cards included messages 1-12; the second set consisted of messages 1, 2, 3, 4, 5, 7, 6, 8, 9, 12, 11 and 10.

Finally, the tapes written by the squoze deck without modifications, including messages 1-12 as data, were compared with the tape written by the squoze deck with modifications. In all cases, the program performed as expected, i.e., when the squoze deck without modifications

was used, a listing was given of the input cards, and the binary number associated with each. When the decks with modifications were used, this listing was deleted. In all cases the card image was printed if a card was numerically out of order. A check of the tape dumps showed the two tapes to be identical.

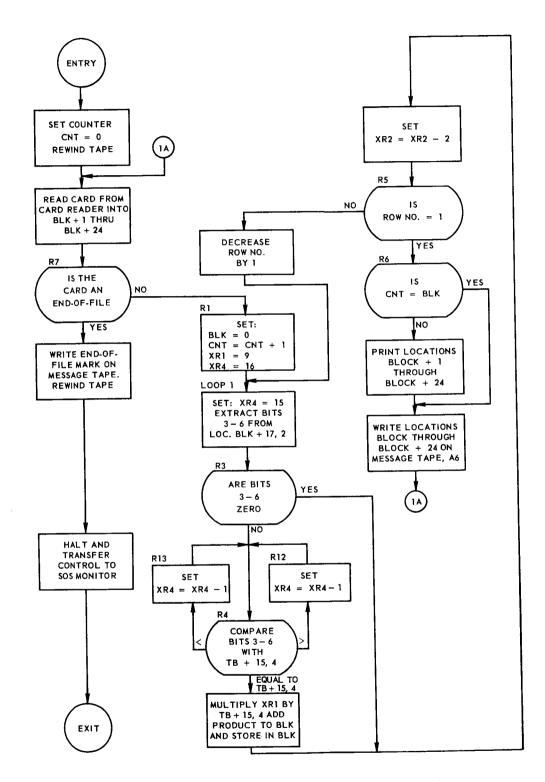


FIGURE 4-11. MXWMOT PROGRAM FLOW CHART

TABLE 4-1. ON-LINE MESSAGES
(Sheet 1 of 8)

MESSAGE NO.	MESSAGE	USED BY
1 to 17	(Station/Radar Type) has begun transmission	MFTTIN
18	Watertown Verlort has begun transmission	MFTTIN
19	Huntsville Verlort has begun transmission	MFTTIN
20 to 22	System tape channel (x) not in ready—next channel being used	MYRSYS
23	This is last vector in orbit table (last vector in orbit table is printed when current time threatens to exceed this table)	MFORMC
24	This is message No. 24	
25	Time to fire reentry table not reach 60,000 ft. This message indicates failure to complete time to fire calculations and the probable absence of a complete set of updated recommended retro times.	MFRARF
26	Sense Switch 2 down, in keys If the address of the keys, as specified in the message, contains zero, all differential correction will be bypassed until Sense Switch 2 is reset. If address of the keys contain an internal station number, the differential correction for that station only will be bypassed.	MPDIFC
27	Error in integration. Previous table requested. (The integration in process has failed to converge and previous table has been requested. It is printed on an error return from the Integration routine.)	MFCPNI
28	Inserted R	MFMAN5
29	Inserted V	MFMAN5
30	Inserted time in hrs. mins. secs.	MFMAN5
31	GMTRC for end of present orbit in hrs. mins. secs. This is the computed time to fire retros (GMT) to impact in a specified area for the present orbit.	MFRARF
32	Manual insertion liftoff in hrs. mins. secs.	MFMANI
33	Manual insertion accepted, number of retros fired	MFMAN2
34	Manual insertion accepted, clock reading read	MFMAN3
35	Manual insertion rejected, enter the message again	MFMAN1
36	Abort table has been generated	MFCPN1

TABLE 4-1. ON-LINE MESSAGES (Sheet 2 of 8)

MESSAGE NO.	MESSAGE	USED BY
37	Abort table reached 60,000 feet	MFCPN1
38	Reentry table did not reach 60,000 feet	MFCPN1
39	Number of observations presented to differential correction by edit is	MFLED1
40	Floating-point trap	MTFLPT
41	Simulated operation begun	MOSENT
42	Normal operation begun	MTWWV1
43	File project tape on B6 and set up new reel	MYSTLT
44	File project tape on B7 and set up new reel	MYSTLT
45	Differential correction rejected On-line messages (45 and 46) will be followed by a number of lines of print indicating the quality of the data that was used in the Differential Correction. If message number 45 is printed without the following lines of print indicating the quality of the data, this indicates an error condition or insufficient input data.	MFDIFC
46	Differential correction successful	MFDIFC
47	T-BDA	MFHOLD
48 to 64	(Station/Radar Tape) has ended transmission	MFTTIN
65	Watertown Verlort ended transmission no. obs.=	MFTTIN
66	Huntsville Verlort ended transmission no. obs. =	MFTTIN
67	Time of apogee in hrs., mins., secs. (This message indicates the time of maximum I R I vector within the next op minutes of Integration Table.)	MFORMC
68	Lamda 0 accepted. Station characteristics tape has been updated	MFMAN 3
69	ACP. TTFW = PA = RW =	MFMANI
70	Lamda 0 accepted. Read in station characteristics tape for updating	MFMAN 3
71	D C rejected. R, V not in table (This indicates that current Integration table does not cover D C interval.)	MFDIFC

TABLE 4-1. ON-LINE MESSAGES (Sheet 3 of 8)

MESSAGE NO.	MESSAGE	USED BY
72	Yes, we have no Cape vector today for pan	MYGEN2
73	Desired impact longitude not between ϕ and 2 PI	MYGEN2
74	Manual insertion accepted landing orbit No.	MFMAN3
75	Longitude (Degrees) accepted	MFMAN3
76	Following station has been deleted from differential correction These messages (76, 77 and 78) are under control of Sense Switch 4 (see Operating Instructions, MC 63-1) and indicate that a manual modification is being made to the Differential Correction processor (D0DIFC)	MFTTIN
77	Following station has been restored to differential correction	MFTTIN
78	Message block for following station is negative	MFTTIN
79	Abort phase above TOWS has been entered	MFLABT
80	Orbit phase has been entered	MFLORB
81	NI Error type use TTF	MFCPNI
82	R, V not in orbit display table at $T = Hrs.$ Mins. Secs.	MFRARF
83	Redundancy occurred on MSG, auxiliary tape being used	MTMFSK
84	Station characteristics tape read successfully	MYSCRD
85	Numerical integration successfully completed	MFCPNI
86 to 109	(Station/Radar Tape) acquisition data sent	MYTTOX MYTTOY MYTTTOX MYTTTOY
110	A record has been written on the restart tape	MTWRRS MYWRRS MTWRSI
111	F1.Pt./Oct Rx = Ry = Rz =	MFCPNI MYREST MFLORB MFLABT

TABLE 4-1. ON-LINE MESSAGES
(Sheet 4 of 8)

MESSAGE NO.	MESSAGE	USED BY
112	Anchor time for above R, V values=hrs. mins. secs.	MYREST MFLORB MFLABT MFCPNI
113	R vector accepted, enter velocity	MFMAN5
114	Velocity vector accepted	MFMAN5
115	Flt. Pt./Oct Vx = Vy = Vz =	MYREST MFLORB MFLABT MFCPNI
116	Note commencement of pass	MSPANS
117	GMTRC display Primary Area is Hrs. Mins. Secs.	MFRARF
118	Apogee, nautical miles (in perigee nautical miles)	MFORMC
119	Lat. long. impact point (longitude is degrees West) This message is printed during an abort or reentry phase only.	MFCPNI
120 to 143	(Station/Radar Tape) acquisition data not sent	MYTTOX MYTTOY MYTTTOX MYTTTOY
144	T-GE	MFHOLD
145	T-IP	MFHOLD
146	Launch Rx = Ry = Rz = These messages (146, 147 and 148) will give the components of the vector used in the launch processors in determining the GO-NO-GO calculation.	MFLHLD
147	Launch $Vx = Vy = Vz =$	MFLHLD
148	R, V not in TTF orbit table at $T = Hrs.$ Mins. Secs.	MFRARF
149	No. of Cape vectors generated forward is	MFCPNI
150	TTF orbit table reached 60,000 feet at Hrs. Mins. Secs.	MFRARF
151	GMTRC Man. insertion area is Hrs. Mins. Secs.	MFRARF

TABLE 4-1. ON-LINE MESSAGES

(Sheet 5 of 8)

MESSAGE NO.	MESSAGE	USED BY
152	Orbit No. of the above vector is	MFDIFC
153	Error in Retrofire program, R6 BOT4 or R6ATAO	MFRARE
154	New reentry table generated	MFCPNI
155	ID frame compare error	MFRARF
156	GMTRC NEA completed	MFRARF
157	Frame sequence error	
158	Line two, low buffer check sum error These messages (158 through 161) indicate IP 7094 input line errors.	10HS09
159	Line two, high buffer check sum error	10HS09
160	Line two, high buffer telemetry rejected	I0HS09
161	Line two, low buffer telemetry rejected	10HS09
162	High-speed output transmission rate to Cape exp., 120, actual	MTWWWV
163	High-speed output transmission rate to Cape exp., 20, actual	MTWWWV
164	High-speed output transmission rate to Cape exp., 10, actual	MTWWWV
165	High-speed output transmission rate to Cape exp., 60, actual	MTWWWV
166	Line one, low buffer check sum error These messages (166 through 169) indicate B-GE input line errors.	IOHSGB
167	Line one, high buffer check sum error	I0HS GB
168	Line one, high buffer telemetry rejected	10HSGB
169	Line one, low buffer telemetry rejected	IOHSGB
170	RSYSERR1 no control or EOF rec after EOF (ETRT2) These messages (170 through 177) indicate that some error has occurred when trying to read the absolute System Tape on tape unit A1.	MYRSYS
171	RSYSERR2, EOF is first rec. read after Rew, spacing files (ETRT3)	MYRSYS
172	RSYSERR3, data rec. first rec. read after Rew, spacing (ETRT3)	MYRSYS

TABLE 4-1. ON-LINE MESSAGES (Sheet 6 of 8)

MESSAGE NO.	MESSAGE	USED BY
173	RSYSERR4,MSREXX err ret-decode file control rec. (ETCWD)	MYRSYS
174	RSYSERR6,MSRECC err ret-decode control rec. after Rew, spacing	MYRSYS
175	RSYSERR7, file no. requested not = file read (ETTS1)	MYRSYS
176	RSYSERR8, MSRECC err ret-decode data rec. (ETECC)	MYRSYS
177	RSYSERR9, data record read too small (ETECC)	MYRSYS
178	Low abort phase has been entered	MFLRT1
179	Medium abort phase has been entered	MFLRT2
180 to 198	Differential correction rejected	MFDIFC
199 to 217	Differential correction successful	MFDIFC
218	Orbit-reentry phase change accomplished	MYREST
219	Time of liftoff is (GMT) in hrs. mins. secs.	MFMANI MFHS08 MFHSGB MFML6A
220	N = 0, edit program rejected, differential correction bypassed N is the number of observations correctly received by the station presently transmitting. If N = 0, this indicates a particular set of low-speed data is not being presented to the Differential Correction program for processing.	MFLED1
221	The WWV time entered is (GMT) in hrs. mins. secs.	MTWWV1
222	Escape rockets fired	MLUPDT
223	Tower separation signal has been received	MLUPDT
224	Abort initiate signal has been received	MLUPDT
225	SECO signal received	MLUPDT
226	Spacecraft separation signal received	MLUPDT
227	Spacecraft separation assumed	MLUPDT
228	No good vectors written on the restart tape	MYRRRS
229 to 232	No, posigrade rockets fired	MLUPDT
233	B-GE IP 7094	MFLHLD

TABLE 4-1. ON-LINE MESSAGES

(Sheet 7 of 8)

MESSAGE NO.	MESSAGE	USED BY
234	BDA	MFHOLD
235	The FDO switch has been changed	MPSARC
236	Summation of $W + N$ for H.S. BD data is	MFLHLD
237 to 249	This is message No. xxx	
250	Neither Restart Tape can be error corrected	MYRRRS MYSRST
251	An erroneous trap occurred on the DCC, subchannel number	MTERTC
252	Display table reached 60,000 ft. at hrs. mins. secs.	MFCPNI
253	Splash* GMT exceeds last time entry in reentry table	MFABRT
254	Reentry table generated based on hrs. mins. secs.	MFCPNI
255	Time restarted from is hrs. mins. secs.	MTWWVI
256	Neither restart tape can be read, insert restart values This message is printed as a result of restarting the tracking program. This would occur in case of machine malfunction. The restart values that should be in- serted via paper tape (see restart operation, MC 63-1) are the time of liftoff and the R, V and T values at time of insertion.	MYSRST
257	Manual insertion accepted orbit switch	MFMAOS
258	Manual insertion accepted abort switch	MFMAOS
259	Manual insertion retrofire time is hrs. mins. secs.	MFMAN2
260	RSYSERA, data record read too large (ETECC) This message indicates that some error has occurred when trying to read the absolute system tape on tape unit A8.	MYRSYS
261 to 264	This is message No. xxx	
265	Manual insertion accepted. Setting is hrs. mins. secs. (Prints the manually inserted spacecraft clock setting.)	MFMAN 3
266	This is message No. xxx	
267	RSYSERR, tape check trap. Machine error if system not enabled This message indicates that some error has occurred when trying to read the absolute system tape on tape unit A1.	MTRSYS
268	Signal to enter orbit phase has been received	

TABLE 4-1. ON-LINE MESSAGES (Sheet 8 of 8)

MESSAGE NO.	MESSAGE	USED BY
269	RSYSERR, Loc. 10, bits 13-17, illegal config. prob. machine error This message indicates that some error has occurred when trying to read the absolute system tape on tape	MTRSYS
070	unit A1.	MYSCRD
270	Station characteristics auxiliary tape cannot be read	MISCRU
271	Signal to enter abort phase has been received	WEU 01 D
272	V avg V go = in ft. per sec.	MFHOLD
273 to 275	Channel (x) in use over 30 seconds. System tapes on next channel being used	MYRSYS
276	AOSTAD output package loaded hrs. mins. secs.	MYRSYS
277	O5ORMC package loaded hrs. mins. secs.	MYRSYS
278	OOORMC package requested hrs. mins. secs.	MYLSYS
279	R5RARF package loaded hrs. mins. secs.	MYRSYS
280	Integration package requested hrs. mins. secs.	MYLSYS
281	D C package loaded hrs. mins. secs.	MYRSYS
282	Edit package requested hrs. mins. secs.	MLYSYS
283	Restart package loaded hrs. mins. secs.	MYRSYS
284	RSYSERRB file requested from SYS tape does not exist. File =	MYRSYS
285	MYREST package loaded hrs. mins. secs.	MYRSYS
286	Inserted Delta - T correction B-GE plus hrs. mins. secs.	MFMAN 4
287	Inserted Delta — T correction B-GE minus hrs. mins. secs.	MFMAN 4
288	Inserted Delta — T correction I P plus hrs. mins. secs.	MFMAN 4
289	Inserted Delta — T correction I P minus hrs. mins. secs. (286 - 289 are printed to indicate the manually inserted corrections being applied to the respective high-speed	MFMAN 4
290	Manual insertion rejected, D.C. in process	MFMAN5

4.13 STATION CHARACTERISTICS TAPE WRITER PROGRAM (U0STCH)

U0STCH generates the station characteristics tape for the Mercury Programming System. This tape consists of a 34-cell block for each station type in the Mercury tracking network—a site with both AN/FPS—16 and Verlort radar units has a separate station characteristics block for each radar. Each block contains the station identification and geodetic, meteorological, and other data with affect measurements of the spacecraft orbital position.

The flow chart for the U0STCH program is shown in Figure 4-12.

4.13.1 Input Requirements

Input to U0STCH consists of a card deck containing the following cards:

- a) Station Number Card—the binary number contained in row 9, columns 1-36 of the first card in the input deck defines the number of station characteristics blocks to be assembled. If, for example, five station characteristics blocks are to be assembled, punches must be present in row 9, columns 34 and 36. Except for the 9-row, this card is blank.
- b) Station Characteristics Tape Identification Card—the second card is in squoze format. It contains 12 words which supply the identification record on the station characteristics tape. All words are in BCD format. Words 1-4 contain the expression STATION CHARACTERISTICS. Words 5 and 6 contain the date, e.g., 7/1/61. Words 7-9 are blanks. Word 10 contains the number of stations to be assembled on the tape. Word 11 is blank. Word 12 contains the number of words in each station block.
- c) Station Characteristics Cards—there must be two row-binary cards for each station to be assembled. If there are less than 25 characteristics, all of the characteristics are located on the first card and a blank card must be added to complete the 2-card set. (NOTE: The cards containing the characteristics may be obtained by a 9 AP assembly of the following symbolic deck: two comments cards, a FUL card, one symbolic card for each characteristic, and an END card. The FUL card eliminates the checksum from the 9 AP punched-card output.)

4.13.2 Output Requirements

Upon completion of UOSTCH, tapes A7 and A8 contain an 18-word identification record and as many other records as there are station blocks. The identification record consists of 12 data words and six error correction words. Each

station block consists of as many words as there are characteristics and ten error correction words. An end-of-file terminates the data on the output tapes.

The contents of the station characteristics blocks, and U0STCH station identification, are illustrated in Tables 4-2 and 4-3 respectively.

4.13.3 Method

The following equations are used to compute the various listed values for the station characteristics blocks:

a) Inertial Longitude of Station at Reference Time (λ_1):

$$\lambda_1 = \lambda + \lambda_0$$

where:

 λ = geodetic longitude (given)

 λ_0 = inertial longtude of Greenwich at reference time (computed from given date).

Source: American Ephemeris and Nautical Almanac, 1960.

b) Geocentric Latitude (φ '):

 φ ' = φ ' - 11'35'' .6635 $\sin 2\varphi$ + 1'' .1731 $\sin 4\varphi$ - 0'' .0026 $\sin 6\varphi$ where φ = geodetic latitude (given)

Source: American Ephemeris and Nautical Almanac, 1960.

c) Altitude Above Ellipsoid in Earth Radii (H):

Hearth radii = (H_{feet} x 0.3048) meters/6378145 meters

Source: Memorandum of Dr. Paul Herget, Cincinnati Observatory.

TABLE 4-2. STATION CHARACTERISTICS BLOCK CONTENTS

Word No.							
1 2 3	Prefix: Site Type (0 = AN/FPS-16; 1 = Verlort; 2 = telemetry) Decrement: Earth Sector Number Address: DCC Input Subchannel Number Prefix: Other radars at this site (7 = AN/FPS-16; 1 = Verlort; 0 = none) Address: Internal Station Number DCC Output Subchannel Mask (Subchannel 10 = PZE 0,0,16; Subchannel 11 = PZE 0,0,8)						
4,5							
	İtem	Source	Source Format	Source Unit	Converted Format	Converted Unit	
6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22	Geodetic Longitude (λ) Geodetic Latitude (ϕ) Geocentric Latitude (ϕ) sin ϕ cos ϕ R_s sin ($\phi-\phi'$) Radius from Earth Center to Sta. (R_s) Altitude above Ellipsoid (H) Local Vertical Deflection Longitude ($\Delta\lambda$) Local Vertical Deflection Calibrated Boresight Elevation Calibrated Boresight Azimuth Inertial Longitude at Reference Time Air Pressure at Ground Level (P_q) Temperature at Ground Level (P_q) Water Vapor Content at Ground Level (P_q) Azimuth Deviation from True North	Given Given Computed Computed Computed Computed Given Given Given Given Computed Given	Dec. Dec. Dec.	DMS* DMS* DMS* DMS* DMS* Millibars C Millibars	Flt. Pt. Flt. Pt.	Radians Radians Radians Radians Radians Earth Radii Earth Radii Earth Radii Earth Radii Radians	
23 24 25 26 27 28 29 30 31 32 33	Square Root of Weight 1 (Range) Square Root of Weight 2 (Azimuth) Square Root of Weight 3 (Elevation) Modulus (n-1) R $\sin (\phi - \phi')$ R $\cos (\phi - \phi')$ $\sin \phi'$ $\cos \phi'$ Boresight Elevation Correction $(C_3)^{**}$ Boresight Azimuth Correction $(C_2)^{**}$ Open for Expansion	Given Given Given Computed Computed Computed Computed Computed	Dec. Dec. Dec.		Flt. Pt.	Earth Radii Earth Radii Radians Radians Radians Radians	

^{*}DMS—degrees, minutes and seconds **Original Input—raw radar reading

TABLE 4-3. U0STCH STATION IDENTIFICATION

Internal Station	Name of Station	Site Type	Earth Sector	DCC S	bchannel	Site Name
Number				In	Out	in Station Block
1	Cape Canaveral	AN/FPS-16	1	14	11	0CPCANAVERAL
2	Grand Bahama Island	AN/FPS-16	1	14	11	0GRANDBAHAMA
3	San Salvador Island	AN/FPS-16	1	14	11	0SANSALVADOR
4	Bermuda	AN/FPS-16	1	17	10	0BERMUDA
5	Bermuda	Verlort	1	17	10	 IBERMUDA
6	Grand Canary Island	Verlort	4	18	11	1CANARIES
7	Muchea	Verlort	10	19	11	1MUCHEA
8	Woomera	AN/FPS-16	11	24	11	0WOOMERA
9	Hawaii	AN/FPS-16	15	25	10	0HAWAII
10	Hawaii	Verlort	15	25	10	1HAWAII
11	Point Arguello	AN/FPS-16	16	20	11	0PNTARGUELLO
12	Point Arguello	Verlort	16	20	11	IPNTARGUELLO
13	Guaymas	Verlort	17	26	10	1GUAYMAS
14	White Sands	AN/FPS-16	17	21	11	0WHITESANDS
15	Corpus Christi	Verlort	18	27	10	1CPUSCHRISTI
16	Eglin	AN/FPS-16	18	22	11	0EGLIN
17	Eglin	Verlort	18	22	11	1EGLIN
18	Bermuda Computer	AN/FPS-16	1	15	11	0BERMUDA
19	Bermuda Computer	Verlort	1	15	11	1BERMUDA
20	Mid-Atlantic Ship	Telemetry	3	23	10	2MIDATLSHIP
21	Kano	Telemetry	5	18	11	2KANO
22	Zanzibar	Telemetry	6	18	11	2ZANZIBAR
23	Indian Ocean Ship	Telemetry	8	24	11	2INDIANOSHIP
24	Canton Island	Telemetry	14	19	11	2CANTONISLES

d) Earth Radius Measured from Center to Mercury Ellipsoid (R):

$$R_{E} = \nu \sqrt{1 - (2 - e^{2}) e^{2} \sin^{2} \varphi}$$

where:

$$\nu = 1/\sqrt{1 - e^2 \sin^2 \varphi}$$

e = eccentricity of earth (computed from e = $\sqrt{595.6}/298.3$, as given in NASA, Appendix A).

 φ = geodetic latitude (given).

e) Earth Radius as Measured from Center to Station (R_s) :

$$R_s = \sqrt{R_E^2 + H^2 + 2\nu H(1 - e^2 \sin^2 \varphi)}$$

where $H = H_{\text{earth radii}}$ and all other terms are as defined previously.

f) Modulus of Refraction (n-1):

$$n-1 = \left[P_g + \frac{77.6 + 4810e_g}{T_g} \right] \times 10^{-6}$$

where:

P_g = air pressure at ground level (given, millibars)

 T_g = temperature at ground level (given, ${}^{o}K$)

eg = water vapor pressure at ground level (given, millibars)

Source: Westerman memorandum of 7/31/58.

g) Boresight Elevation Correction (C3) and Azimuth Correction (C2):

$$C_2 = A_a - KA_r$$

$$C_3 = E_a - KE_r$$

where:

A_a = calibrated boresight azimuth (given)

E_a = calibrated boresight elevation (given)

 $A_a = azimuth reading (given)$

 E_r = elevation reading (given)

K = radian conversion factor

Source: Mr. James Donegan of NASA.

4.13.4 Usage

a) Calling Sequence—U0STCH is an independent program but may be used as a subroutine with the following calling sequence:

Location	<u>Operation</u>	Address, Tag, Decrement
Alpha	T TSX	UOSTCH, 4
Alpha + 1		Normal return

- b) Error Conditions—two SOS library utility programs, U1SQRT and U1SICO, are incorporated into U0STCH as subroutines. An error return from either of these programs results in a program halt.
- c) Storage Required-639 locations.
- d) Accuracy-26 significant bits.

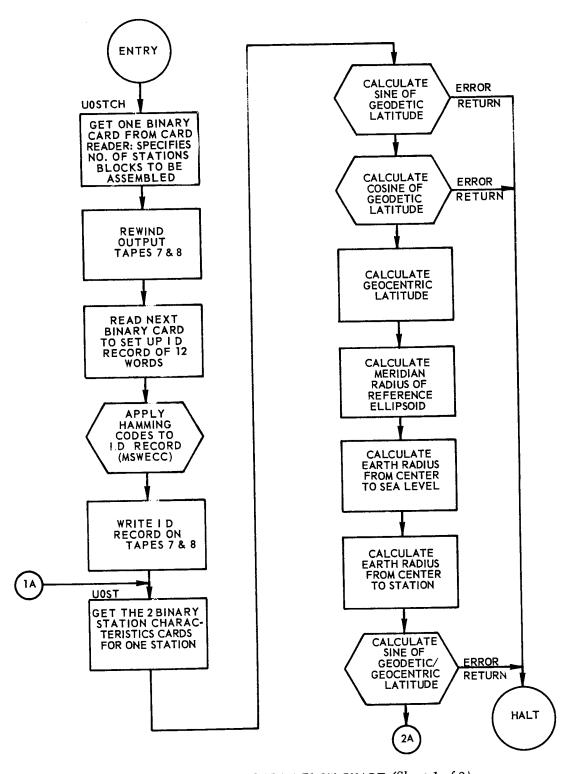
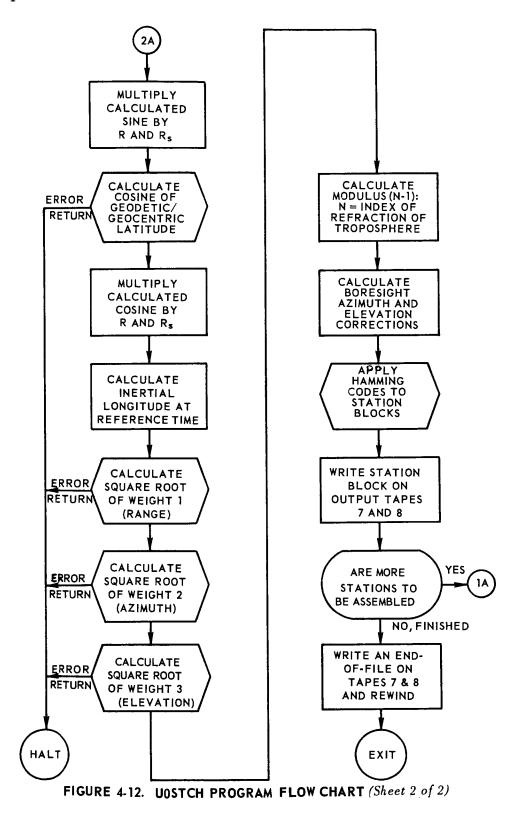


FIGURE 4-12. U0STCH PROGRAM FLOW CHART (Sheet 1 of 2)



4.14 STATION CHARACTERISTICS TAPE UPDATING PROGRAM (UOSTUP)

After U0STCH has generated the station characteristics tape, U0STUP is used to insert and/or delete station blocks, delete one or more characteristics from every block, or alter various characteristics in individual blocks. Updating the station characteristics tape to correspond to the latest available information is part of the countdown procedure for a Mercury mission.

A general flow diagram and a detailed flow chart for the U0STUP program are shown in Figures 4-13 and 4-14, respectively.

4.14.1 Input Requirements

Input to U0STUP consists of the following items:

- a) The station characteristics tape to be updated on A7 and a duplicate tape in reserve on A8. These tapes are the output from U0STCH (see subsection 4.13.2).
- b) A card deck of the updating corrections—the first card is an on-line print control card and is blank except for row 9—a punch in column 30 produces an on-line printout of all station blocks prior to updating; a punch in column 60 produces an on-line printout of all blocks subsequent to updating.

The second card contains the inertial longitude of Greenwich (λ_0) at the reference time. The inertial longitude is given in units of time—hours beginning in column 1, minutes beginning in column 10, seconds beginning in column 20, in Hollerith.

The third card contains the date of λ_0 , coded in Hollerith, e.g., 07/01/61. The remaining input cards contain Hollerith-coded corrections to the station characteristics blocks (see Tables 4-4 and 4-5).

c) When inserting a new station number, station numbers following the new entry must be altered to maintain consecutive numbering. For example, two new stations inserted between existing stations 5 and 6 and 8 and 9 cause internal station numbers to be changed as follows:

internal station numbers 5 6 7 8 9 10 1

TABLE 4-4. FORMAT, STATION CHARACTERISTICS CHANGES

	Card Format			
Туре	Start in Column 1	Start in Column 10	Start in Column 20	
1 Normal Change	Station Number	Char. No. 32 must follow 17, and 33 must follow 18, for the same station	Updating data for an angle: Col. 20: deg (dec. int.) Col. 30: min (dec. int.) Col. 40: sec (mixed number) (If negative, a minus (-) sign in cols. 20, 30 and 40) For Pg, Tg or eg*, only one card is needed to update all stations at one site. For characteristics 1, 2, 3, 4, 5, 14, 20, 21, 22, 24, 25, 26, 32 and 33, data begins in column 20.	
Deletion of Characteristics from All Stations	Blank	Characteristic Number	Blank	
3 Deletion of a Station	Station Number (most recent)	Blank	Blank	
4 Insertion of a Station	Blank	Blank	Station Number	

^{*}Pg—Air Pressure at Ground Level

 T_g —Temperature at Ground Level

eg—Water Vapor Content at Ground Level

TABLE 4-5. CARD SEQUENCE, STATION CHARACTERISTICS CHANGES

Group Type Number		Remarks		
1	1	One for each characteristic for each station		
	2	Delete a characteristic from all stations		
2 (may be repeated)	1	A new characteristic, if any, for any or all stations, in place of the deleted characteristic above.		
	3	Delete a station, if any.		
3	4	Insert a station, if any.		
(may be repeated)	1	One for each characteristic for this new station.		

No card should follow the last updating data card.

4.14.2 Output Requirements

- a) On-line Printout:
 - 1) Heading, STATION CHARACTERISTICS
 - 2) The value, λ_0 , in hours, minutes and seconds
 - 3) Date of λ_0
 - 4) Station blocks, before updating, in floating-point decimal form (optional)
 - 5) Updating station characteristics (input card format)
 - 6) The value, λ_0 , in floating-point decimal
 - 7) Station blocks after updating in floating-point decimal (optional). A change on a printout is indicated by an asterisk (*)
 - 8) Ending sentence

U0STUP provides on-line printouts when specific types of errors are detected in the execution of the program.

b) Output Tapes B9 and B10-identical updated station characteristics tapes.

4.14.3 Method

The equations used by U0STUP in updating the station characteristics tape are identical to those used by U0STCH in generating the original station characteristics tape, except for the equation used to compute the geocentric latitude (ϕ') :

$$\varphi' = \arctan \left[(1 - f)^2 \tan \varphi \right]$$

where:

 φ = geodetic latitude

f = flattening, (1/298.3)

4.14.4 Usage

a) Calling Sequence—U0STUP is an independent program but may be used as a subroutine with the following calling sequence:

Location	Operation	Address, Tag, Decrement
alpha	TSX	U0STUP, 4
alpha + 1		Normal return

- b) Error Conditions—two SOS library utility programs (UISQRT and UISICO), an error correction code writing program (MSWECC), an error correction code reading program (MSRECC), two format conversion programs (BCD1 and FILE), a card reading program (RCD1) and an on-line writing program (TOPC2) are all incorporated into U0STUP as subroutines. An error return from any of these subroutines results in an on-line printout indicative of the error.
- c) Storage Required-6457 locations.
- d) Accuracy-26 significant bits.

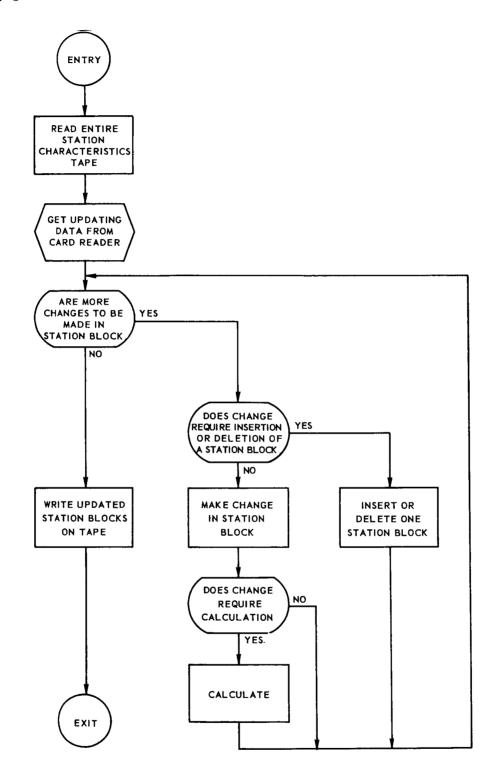


FIGURE 4-13. UOSTUT GENERAL FLOW DIAGRAM

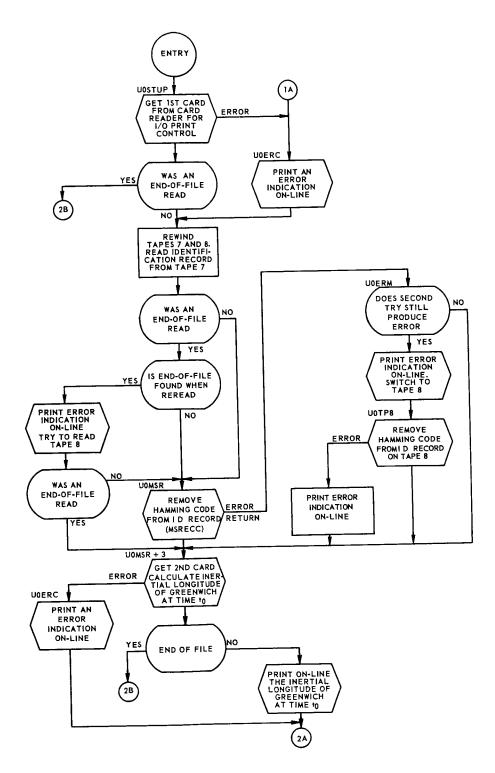


FIGURE 4-14. U0STUP PROGRAM FLOW CHART (Sheet 1 of 7)

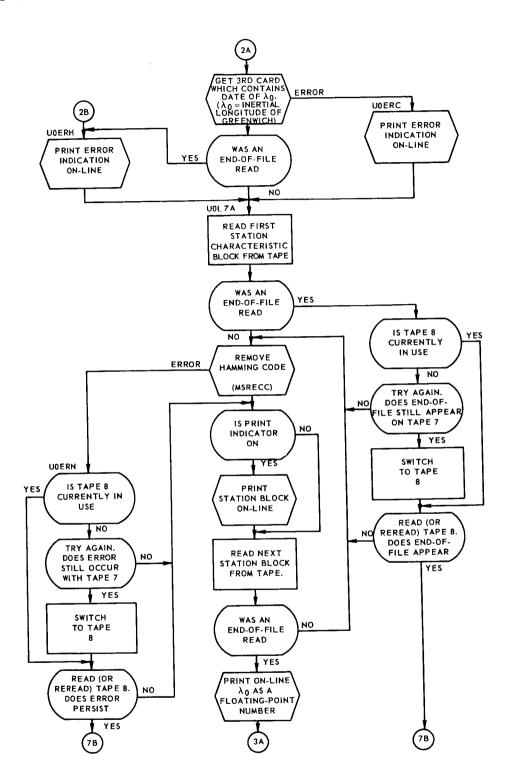


FIGURE 4-14. UOSTUP PROGRAM FLOW CHART (Sheet 2 of 7)

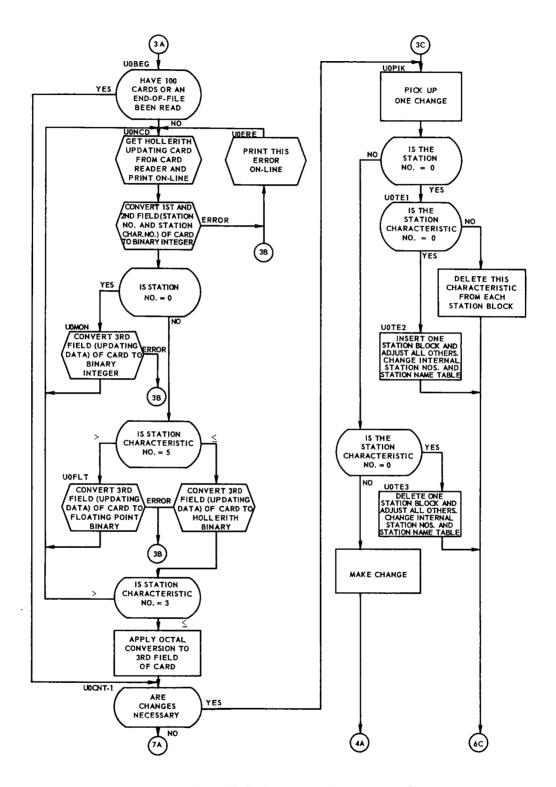


FIGURE 4-14. U0STUP PROGRAM FLOW CHART (Sheet 3 of 7)

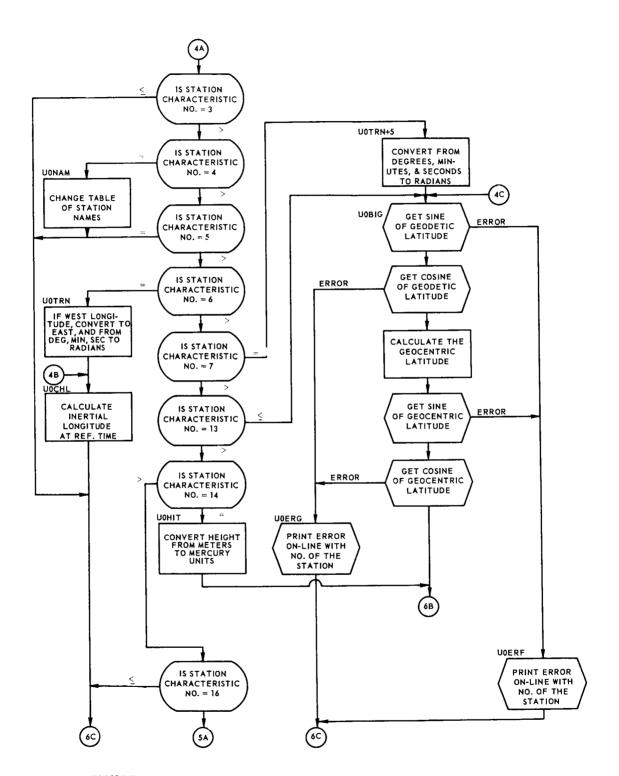


FIGURE 4-14. UOSTUP PROGRAM FLOW CHART (Sheet 4 of 7)

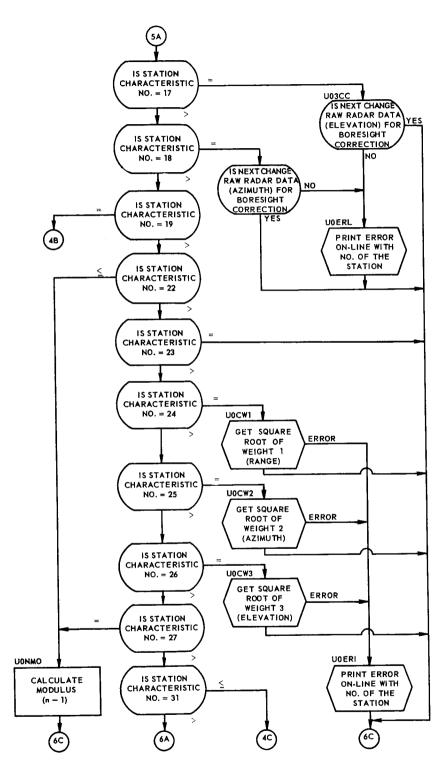


FIGURE 4-14. UOSTUP PROGRAM FLOW CHART (Sheet 5 of 7)

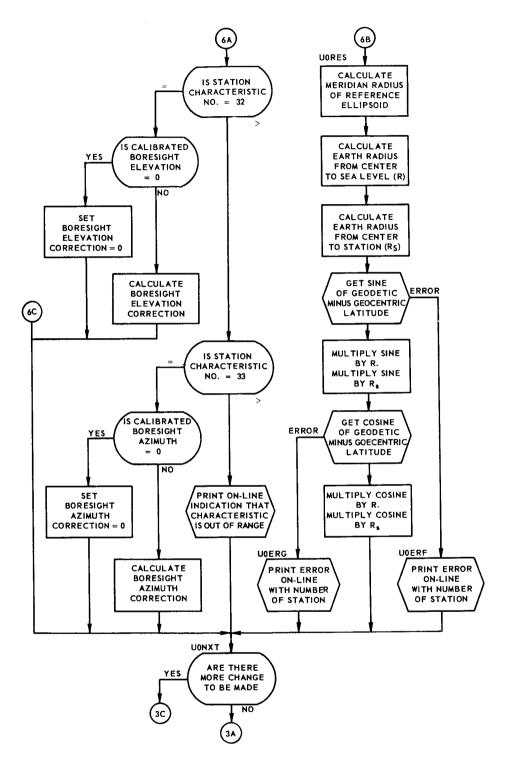


FIGURE 4-14. UOSTUP PROGRAM FLOW CHART (Sheet 6 of 7)

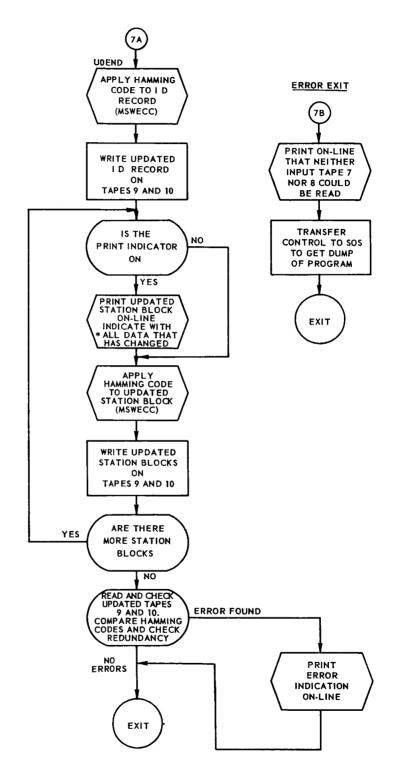


FIGURE 4-14. UOSTUP PROGRAM FLOW CHART (Sheet 7 of 7)

Section 5

POSTFLIGHT ANALYSIS AND REPORTS

Each time the Mercury Operational System is run, whether on an actual mission or a simulated mission for test, a log tape is made. The log tape contains all data produced during the mission—input and output between Goddard, Cape Canaveral, and the worldwide tracking network—and is analyzed after the run to record important data and information about the mission.

The postflight reports are generated by the self-contained postflight reporter program written in Fortran. The Postflight Reporter program recalculates certain parameters and calculates other parameters which either were never calculated by the operational program (such as aerodynamic parameters) or which were calculated but were never recorded (for example, longitude of node). Being self-contained, the program contains a monitor program (presented in subsection 5.5) and a collection of other programs which is divided into four categories:

- a) Initialization Programs—presented in subsections 4.6 to 4.9, establish certain data and values to be used by subsequent programs.
- b) Tape Processor Programs—presented in subsections 4.9 to 4.22, process the tapes and extract the necessary information.
- c) Phase Processor Program—produce the data for the reports. These programs and their support programs are discussed in subsections 4.22 to 4.35.
- d) Utility Programs—presented in subsection 4.35.

Subsection 4.36 presents the program operating procedures and Appendix A defines the symbols used by the Postflight Reporter program.

Additional information pertinent to the Postflight Analysis and Reports section is given in Appendixes B, C, and D, which are titled Postflight Reporter Symbolic Designations, Coordinate Conversion Systems, and Report Data Formats, respectively.

5.1 POSTFLIGHT MONITOR PROGRAM

Postflight Monitor is a control program developed specifically for the postflight reporter. It makes logical decisions and directs the functioning of the various subroutines and is in no way related to the Mercury Programming System Monitor. Its operations directly produce the postflight reports. The Postflight Monitor supervises the search for output from which corresponding input is derived, which is the reverse of the procedure employed in the Mercury Programming System.

Monitor relies on external and internal controls. Internally, it uses COMMON to facilitate the exchange of data among its subroutines. COMMON creates storage for the general interchange of information. Externally, Monitor requires the use of control data cards, sense switches, and entry keys to exercise the various options of the program.

The general flow chart for the Postflight Reporter program is shown in Figure 5-1. The flow chart for the Monitor program is shown in Figure 5-2.

5.1.1 Input Requirements

The 12 major subprogram inputs to the Monitor program are:

- a) CHUMLY-sets up BCD output blocks in core.
- b) ACTORS—sets conversion constants into core.
- c) INITIA—initializes system constants and geophysical parameters.
- d) SORTER—sorts the log tape into high-speed input/output data and determines discrete event timing.
- e) GETME-supplies times of discrete events.
- f) DONOUT-obtains high-speed output data.
- g) DONIN-obtains high-speed input data.
- h) LAUNCH-calculates launch and abort parameters from high-speed input data.
- i) ORBIT-calculates orbit parameters from orbit display output data.
- j) RENTER—calculates abort and reentry parameters from the reentry display output data.
- k) NUMIN-calculates a series of position and velocity vectors by use of the numerical integration program.

- 1) GRUNGY—interpolates for a pair of position and velocity vectors at a given time by use of the Langrangian interpolation routine.
- m) DISTAN-computes distances traveled by using geocentric latitude and longitude.

These major inputs use a number of subordinate programs which are discussed later in this section.

Another source of input to Monitor is COMMON which contains all constants and parameters. COMMON is a block in high-order core storage of 175 locations. It begins at 77461₈ and proceeds downward in memory until the COMMON list is exhausted. It includes some dimension quantities. The composition of COMMON is described in subsection 4.5.4 which also lists the Fortran routines that must be in core for successful operation of Monitor and its subprograms.

5.1.2 Output Requirements

All output from the Monitor program is placed in COMMON and may be printed on-line and written onto tape. The actual Monitor output data is used to write the postflight reports. The program investigates four flight phases—launch, abort, orbit, and reentry—for stated periods of time and selects the data required for the specified reports. In addition, discrete event occurrences are reported.

At the beginning of each phase of both the Quick Look and Three Day Reports, a heading is printed containing the name of the phase, the Greenwich Mean Time of liftoff, and the longitude of Aries at liftoff, in degrees. The heading information is printed one time only for each applicable phase. The paragraphs following the headings contain the results of the postflight investigation. (The headings and data formats are detailed in Appendix C of this manual.)

In the Condensed Report, the heading consists of the name of the phase and the data format of the parameters. This heading is printed at the beginning of each page of the report. The Condensed Report can be suppressed by depressing sense switch 3.

5.1.3 Method

This subsection discusses the manner in which Monitor organizes and controls the flow of information to produce the postflight reports.

At the start, Monitor calls the initialization program CHUMLY (see subsection 4.6) to read into core three BCD output blocks. It then calls ACTORS (see subsection 4.7) to place a series of constants into a common block called FACTOR. On the return from ACTORS, the Monitor prints a series of headings and data formats (see Appendix C) with on-line option. These headings and data formats actually contain the key for the postflight report.

Monitor then reads in one card from the on-line card reader, which contains two required physical constants*, and calls INITIA (see subsection 4.8). The INITIA Program initializes a series of physical constants and converts them to forms more useful to Monitor. After returning from INITIA, the main program prints on-line: IF SORT IS NECESSARY - ENTER THE NUMBER OF PHYSICAL LOG TAPES IN THE ADDRESS OF THE KEYS AND PRESS START. IF SORT IS TO BE SKIPPED - DEPRESS SENSE SWITCH 6, MOUNT A4 AND B4, AND PRESS START. SENSE SWITCH 1 MAY BE TOGGLED FOR ON-LINE GLIMPSES OF C3 OUTPUT. DEPRESS SENSE SWITCH 3 TO SUPPRESS CON-DENSED REPORT ON B2. DEPRESS SENSE SWITCH 2 TO SUPPRESS DIS-TANCE COMPUTATION, OUTPUT ON A-3. The program comes to a PAUSE with 77771_{Q} in the address portion of the storage register. After the operator presses START, the program tests sense switch 6 to determine whether the program will proceed to SORTER (see subsection 4.9). If the switch is up, a sort is done; if it is down, the sort is suppressed. Although the output of SORTER is necessary to Monitor, it need not be redetermined once obtained.

On return from SORTER, the program determines whether the sort was successful. If the routine was unsuccessful, a message states that AN ERROR HAS BEEN DETECTED IN KEY OR TAPE SET-UP. PLEASE REVIEW OPER-ATING NOTES. AFTER COMPLETING CORRECTION PRESS START. The program, which is at PAUSE 77775₈, attempts to resort. If successful, the program is ready to continue. When no sort can be made, the program can go no further.

Monitor then calls the GETME subroutine (see subsection 4.19). GETME searches a tape** created by SORTER for the occurrence of seven discrete events: GMTLO; SECO (or abort initiate); time of inception of abort or orbit phase, and of reentry phase; number of retrorockets fired; and time first retrorocket fired. The time of occurrence of the last six events are all referenced to liftoff (GMTLO).

Monitor returns from GETME and tests an indicator to determine whether liftoff has been found. If it has not been found, Monitor assumes that there has been a setup error. The operator is instructed to review his operating notes and correct the error. There is a PAUSE at 77773₈, and when the operator presses START, Monitor returns to GETME. If the GMT of liftoff (GMTLO) is not found, Monitor cannot continue.

Using GMTLO, Monitor computes the Greenwich hour angle of Aries at lift-off and transfers to FIXIT1 which ensures that the angle lies between 0 and 2 π

^{*} The constants read in are THETAO, a launch day constant, and the name of the computer used. (See Appendix A for Fortran symbols and definitions.)

^{**} The tape created by SORTER is a high-speed input message tape. GETME searches the second file of this tape.

radians (including zero). Monitor converts this to degrees and transfers to FIXIT which ensures that this angle lies between zero and 360 (including zero).

Monitor tests whether the log tape contains the occurrence of six discrete events (all events searched for by GETME except the number of retrorockets fired). If any of the six events have not been found, the program will continue with a zero value for each event. If the events have been found, HRSCNV is called to convert these times from floating-point seconds to fixed-point integral hours, minutes, and seconds. FIXIT2 ensures that the hours lie between 0 and 24 (including zero). Monitor prints on-line and writes on C3 the time of occurrence of the individual event or the fact that it has not been found.

After initialization has been completed, Monitor is ready to process the data and write its report. A request-for-report data card is read from the online card reader. This card indicates the type of report data that is requested. A zero in column one of the card indicates that the job is complete and that all data requested from the particular log tape is obtained. In this case, Monitor writes on C3, the output tape, that the end of the job has been reached. An end-of-file is also written on C3, A3, B3, and B2; the C3 tape is rewound and unloaded; and the program comes to a PAUSE at 777778. If the operator presses START, the program transfers to the beginning to read and process a new log tape.

The first character of the request-for-report card, if not a zero, should be a one or a two. Two indicates a request for the Quick Look; one indicates a request for the Three Day Report. The main program sets certain control values positive, converts the start and end times on the input card to seconds, and tests whether the time difference between the first and last times to be read is non-negative. If it is negative, the program prints on-line ERROR IN DECK SET-UP. TF EXCEEDS TL. REPUNCH, PRESS START. There is a PAUSE at 121218, and when the operator presses START, the program transfers back to read in the card again.

If the difference between the first and last times is positive, Monitor prints out a heading indicating the information that is included in the tapes created by SORTER. Monitor also determines the number of observations to be taken from these tapes and the phase which is to be examined. The GMT of the first message is computed and the program prints on-line the GMT of liftoff and the longitude of the first point of Aries at liftoff. Monitor also computes the angle at 2-inch liftoff in the equatorial plane between Aries and the longitude of the GE radar, and the angle in the equatorial plane between Aries and the longitude of the pad.

At this point, Monitor prepares to enter the loop that starts writing the report. DONOUT is called and the high-speed output logged messages are read from the message tape A4 (or A5). The data is removed according to a predetermined time and flight phase.

There are three returns that can be indicated in a parameter, JERROR, of DONOUT's call statement. Indicator one indicates no more output on the log tape for the phase requested; the loop is terminated, and the next data card is read. Two is the normal return. Indicator three shows a redundancy record. In this case, Monitor skips this particular record and proceeds to the next time interval. Monitor saves the time produced by DONOUT (in general, this is the vector time from the log tape) to be incremented later in the loop.

When the indicator of DONOUT is either 2 or 3, Monitor determines the phase being reported. If the phase is launch or abort, Monitor either transfers to DONIN or acts in the same manner as in the error return case of DONIN (see below). However, with processed data available, DONIN is used. DONIN processes the high-speed input message tape. In its call statement are the vector time, the data source, and an error indicator. When control is returned to Monitor and if the error indicator shows that DONIN cannot find the associated input message, the on-line printer states that THE PROGRAM HAS FAILED TO LOCATE INPUT MESSAGES FROM _______ TO ______ TO CORRESPOND-ING OUTPUT MESSAGES.

If DONIN has not found the associated input message, Monitor sets to zero a number of parameters* which normally would be calculated by DONIN. Monitor then turns to core to utilize data deposited there by DONOUT. Depending on the type of report required, the program selects and prints out the appropriate DONOUT data.

If DONIN has found the associated input message, subroutines IPCNV or GECNV are used to convert the coordinates of position and velocity in the appropriate data source reference frame (IP 7094 computer or B-GE computer) to true inertial coordinates (see Appendix B). The LAUNCH program is called, and it produces the calculated output required for the report.

When LAUNCH is entered and there is associated high-speed input, its duties may be divided among reporting launch, abort and reentry displays. There can be associated high-speed input for an MA launch, an MR abort and an MA reentry. If a reentry switch is on, most of the LAUNCH program can be used for the MA reentry situation. If an abort switch is on, RENTER, the reentry processor, must be used for an MA abort situation (there is no associated high-speed input).

After it has produced the required output, LAUNCH returns to Monitor which converts these outputs to their proper units for reporting. A heading is prepared and records are written. If the phase is launch, the records are written according to the Launch Data Format in Appendix C. Detail, if required, is included in the report.

^{*} These parameters, which are defined in Appendix A, are: X, Y, Z, X1, Y1, Z1, U, V, W, U1, V1, W1, VI, PSII, CL, VE, GE, PSIE, ASUBR, S, XMACH, QD, RN, and QS.

Monitor determines whether the condensed report information is also to be written. Sense switch 3 must be up to write the report; if the switch is down, no condensed report will be generated. If the switch is up, a new launch paragraph is written for each 5-second increment of the phase, immediately after Monitor calls RCACNV. This subroutine converts the recovery area to an alphanumeric code. If an abort situation is represented, the procedures are identical to those in launch except that the paragraphs are written in increments of 10 seconds. Time is incremented and Monitor returns to DONOUT to pick up the next message. (This completes one pass through the loop.)

If the phase is orbit, Monitor reads in a time, position and velocity vector. This vector is fed into the NUMIN subroutine which produces a table of position and velocity vectors at one-minute increments over the valid period of the input vector. The number of output points during the period of the input vector is then computed and a loop entered which obtains the data for one of these points on each pass through the loop.

Upon entering the loop, Monitor transfers to the DONOUT subroutine which obtains the high-speed output data for the time nearest the desired point. To obtain a position and velocity vector corresponding to the high-speed output data, Monitor calls the GRUNGY subroutine. This subroutine interpolates for the desired vector within the table of vectors generated by NUMIN. Having acquired both the input and output data, Monitor transfers to the ORBIT subroutine. This program computes the rest of the desired orbital parameters. Upon return to Monitor, the orbital parameters are converted to the desired output units and the output report for that point is then written.

If the Condensed Report is to be generated on B2, Monitor calls subroutine RCACNV and a new orbit paragraph for each additional one-minute increment is written.

The loop described above is repeated until all desired points over the input vector have been computed. When this occurs, Monitor tests to see if the vector just operated on is the last vector. If it is not, the next vector is read in and the whole process is repeated. If it is the last vector, control is transferred back to read in the next phase card.

If the phase is reentry, the procedure is the same as the one used for ORBIT. The only difference is that instead of transferring to the ORBIT subroutine, Monitor transfers to RENTER.

After each phase is processed, a new phase card is read until a data card with a zero in column one is read. As mentioned earlier, this is the end card. The message, JOB COMPLETE, is written on G3; an end-of-file is placed on the output tapes C3, A3, B3, and B2; C3 is rewound and unloaded; and the program comes to a PAUSE at 77777_o.

At this point, another logical log tape may be processed without reloading the program into core storage. If this is desired, the new C3 tape is mounted and START is pressed. If this is not desired, the job may be completely terminated.

5.1.4 Usage

a) Storage—the Postflight Monitor and its subprograms use shared storage locations, COMMON, for all input and output. The contents of COMMON are arranged in the following order:

X, Y, Z, X1, Y1, Z1, U, V, W, U1, V1, W1, XIP, YIP, ZIP, XIP1, ZIP1, YI1, XX1, YY1, ZZ1, XI, ETA, ZETA, XI1, ETA1, ZETA1, VI, VE, GI, GE, PSII, PSIE, RBAR, RRBAR, RADIUS, ARBAR, AXIS, HE, HA, DTHTAP, R01, XLRHO, GMTLO, BCAC, RPBAR, EQURAD, FFLAT, GRAVIT, XMUE, CANJ2, CANJ3, CANJ4, AC, OMEGAE, XLPAD, XLM1, HPAD, XLMO, XNUA, THETA0, DL, CL, XL, RL, ASUBR, XMACH, QD, RN, S, D, CR, SR, QS, VIVR, VIVRGE, VIVRIP, GIGE, GIIP, ECC, XINC, ARGP, ARGP1, OME, OME1, TP, TA, EA, XMA, PER, PHIMIN, XLMMIN, PHIMAX, XLMMAX, PHIIP, XLMIP, DLMI, CS, DENS, XNU, P, NUMORB, NORBCP, XLAMP, DPHIR, JAREA, NOGOGO, DTR, EGT, ECTRS, GTRS, GTL, XICTRC, GMTLC, GMTRC, GMTRC1, GMTRC2, GMTRC3, ECTRC, ECTRC1, ECTRC2, ECTRC3, GMTRS, T, TSECO1, NORET, TRETRO, KHR, KMIN, KSEC, FACTOR, TM, HB, XLM, RO.

Included in the listing of COMMON are four dimension quantities which involve more than one location. They appear below with their respective dimensions.

KHR - 10 KMIN - 10 KSEC - 10 FACTOR - 25

b) FORTRAN Programs—the Postflight Reporter is executed with the Fortran 32K Monitor. In addition to the Fortran Monitor, the following Fortran programs must be read into core for the successful operation of the main program and its subprograms:

(FPT), (CSH), (SLI), (RTN), (STH), (SLO), (FIL), (SPH), (EFT), EXP(2, EXP, SQRT, EXP(3, ATAN, SIN, COS.

For column binary operation, the Fortran Monitor automatically supplies these. For row-binary (absolute) operation, these must be a part of the operational program.

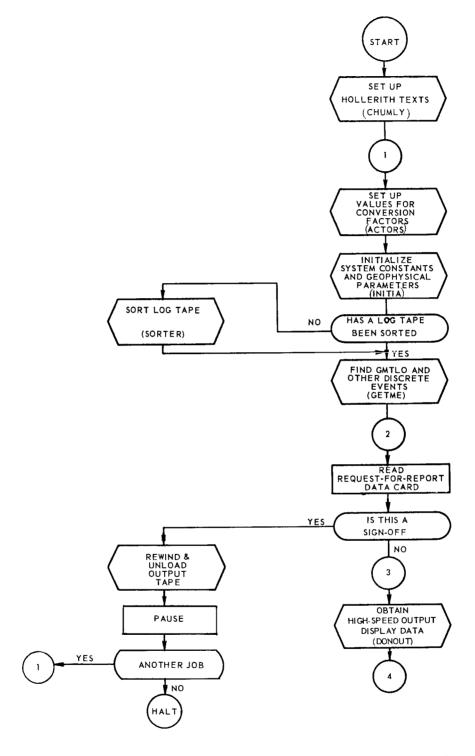


FIGURE 5-1. POSTFLIGHT REPORT PROGRAM, GENERAL FLOW CHART (Sheet 1 of 2)

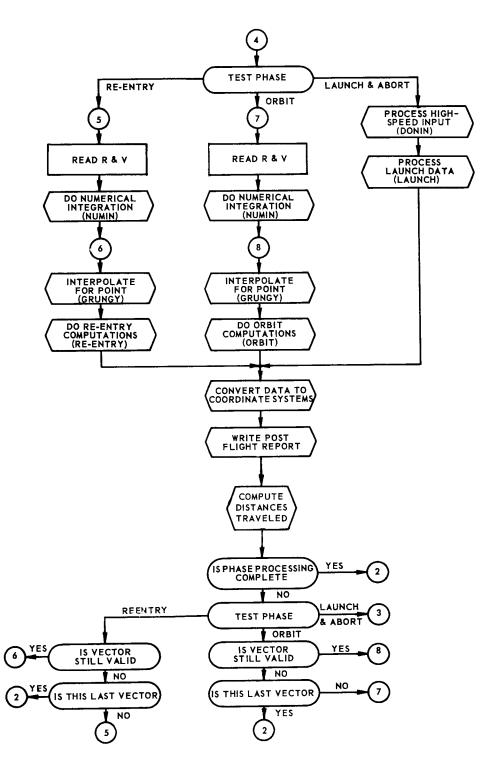


FIGURE 5-1. POSTFLIGHT REPORTER PROGRAM (GENERAL FLOW CHART) (Sheet 2 of 2)

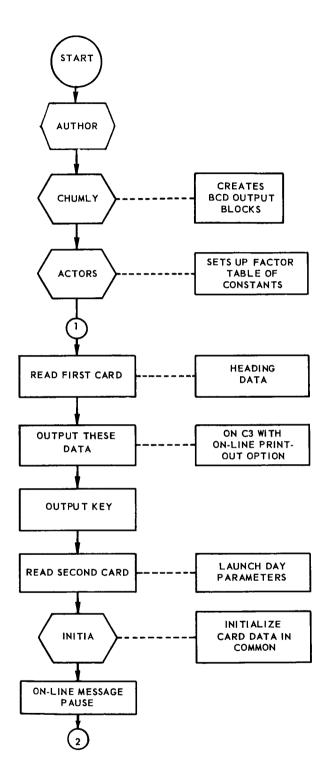


FIGURE 5-2. POSTFLIGHT MONITOR FLOW CHART (Sheet 1 of 4)

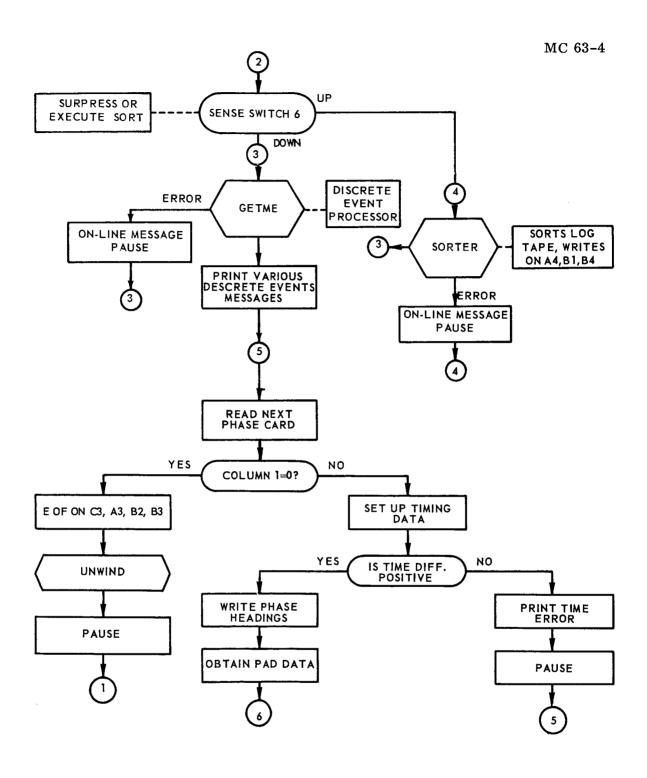


FIGURE 5-2. POSTFLIGHT MONITOR FLOW CHART (Sheet 2 of 4)

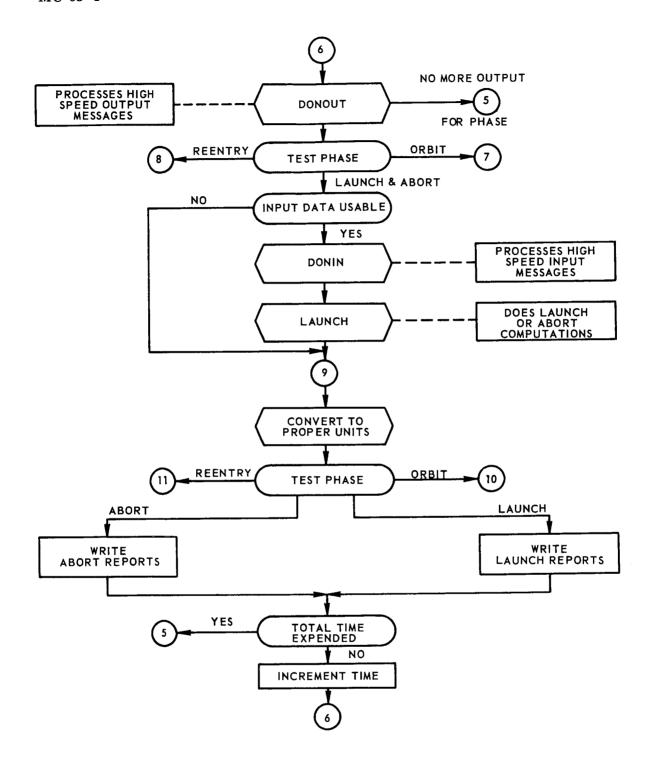


FIGURE 5-2. POSTFLIGHT MONITOR FLOW CHART (Sheet 3 of 4)

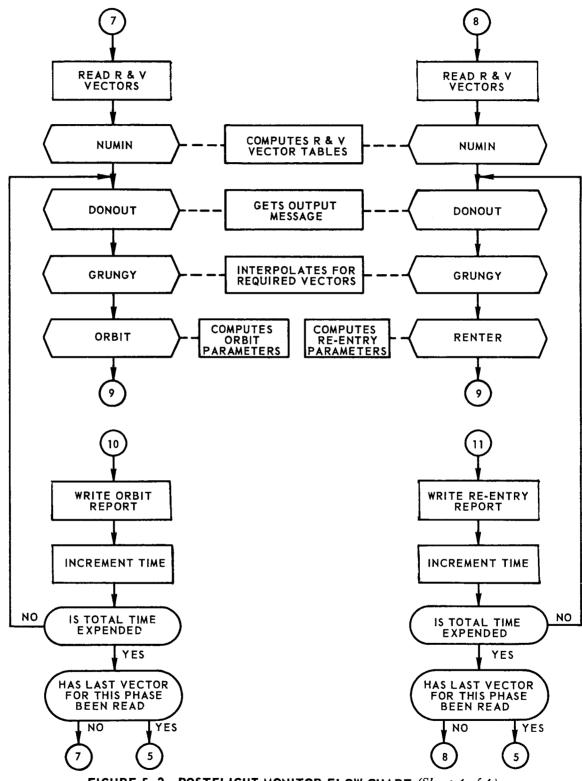


FIGURE 5-2. POSTFLIGHT MONITOR FLOW CHART (Sheet 4 of 4)

5.2 BCD OUTPUT INITIALIZATION PROGRAM (CHUMLY)

CHUMLY creates four BCD output blocks for the Monitor program. The contents of these blocks are read into core storage to provide the BCD information needed for the report. The blocks are:

HED1 Phase Title (Launch, Abort, Orbit, Reentry)

HED2 Data Source Title (B-GE, IP 7094, Raw Radar)

HED3 GO-NO-GO Title (GO-NO-GO, GO and NO-GO, NOT COMPUTED)

The flow chart for CHUMLY is shown in Figure 5-3.

5.3 CONSTANT FACTORS INITIALIZATION PROGRAM (ACTORS)

ACTORS reads into core the constant values of certain conversion factors used throughout the Postflight Reporter Program. These constants read into a common block, FACTOR, are:

FACTOR (1) = Feet per statute mile (5280)

FACTOR (2) = Feet per nautical mile (6076.1155)

FACTOR (3) = Seconds per Mercury unit of time (806.8104)

FACTOR (4) = Kilograms per meter³/slugs per foot³ (515.378725)

FACTOR(5) = Meters per foot(0.3048)

FACTOR (9) = Two pi (6.2831853072)

FACTOR(10) = One radian(57.2957795 degrees)

FACTOR (19) = One pi (3.141592)

5.4 SYSTEM PARAMETER INITIALIZATION PROGRAM (INITIA)

INITIA initializes and converts system parameters required on launch day into the basic internal units needed for use in subsequent programs of the Postflight Reporter. The parameter values obtained by INITIA are:

Time since liftoff (in floating-point seconds) of start of orbit phase

Time since liftoff (in floating-point seconds) of start of abort phase

Time since liftoff (in floating-point seconds) of start of reentry phase

Geocentric latitude of spacecraft at end of retrofire Longitude of spacecraft at end of retrofire Local earth radius of spacecraft at end of retrofire Cavonical equatorial radians in meters Earth flattening Gravity in meters/sec² Earth's gravitational constant in (meters $^3/\text{sec}^2$) x 10^{-14} 2nd harmonic potential in (meters $^{5}/\text{sec}^{2}$) x 10^{6} 3rd harmonic potential in minus (meters $^6/\text{sec}^2$) x 10^6 4th harmonic potential in (meters $^7/\mathrm{sec}^2$) x 10^6 Clarke spheroid equatorial radius in feet Polar radius in feet Radial distance of spherical earth in feet Angular rotation of earth in rad/sec Geodetic latitude of pad in degrees Longitude of pad in degrees Height of pad above mean sea level in feet Longitude of GE central radar in degrees Geodetic latitude of launch pad in radians (28.4908729/2 π) = XLPAD Geocentric latitude of launch pad in radians $\left(\arctan\left(\frac{20854892}{20925832}\right)^2\right)$ tan (XLPAD) = XLRHO Geocentric radius at pad in feet $\left(20925832. / \sqrt{\frac{\cos^2{(\text{XLRHO})} + \sin^2{(\text{XLRHO})}}{(20854892. / 20925832.)^2}}\right)$

= RPBAR

Total distance from geocenter to pad in feet (RPBAR + 11.571) = R01 Longitude of launch pad in radians (-80.547114/2 π) = XLMI Longitude of GE radar in radians (-80.581731/2 π) = XLMO Launch azimuth in radians (THETA0/2 π) = THETA0 Greenwich hour angle at midnight preceding launch in radians Zeroth, second, third and fourth harmonics of earth's potential in English units

NOTE: feet = international feet seconds = mean siderial seconds

INITIA also sets up eight coefficients of each seventh degree polynominal fit (high and low altitude fits) of atmospheric density. The coefficients, determined by a least squares criterion, are placed in a common block for use by the program ATMOS.

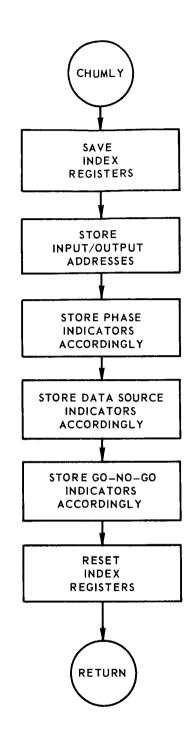


FIGURE 5-3. CHUMLY PROGRAM FLOW CHART

5.5 LOG TAPE SORT PROGRAM (SORTER)

The SORTER program consists of a control program and nine subroutines. SORTER performs the unpacking and processing of the logical log tape and, using the Mercury Programming System log tape(s), creates at least three other tapes as output.

The flow chart for SORTER is shown in Figure 5-4.

5.5.1 Input Requirements

Input to SORTER is the log tape on B6 (and B7 if necessary). The other input is the key setting giving the number of physical log tapes which comprise the logical log tape. The following subprograms are used with this routine: GEB, IPORR, MANIN, HSOP1, BCTB (BCTB1), BCTBI (BCTBJ), TISWS, HMSTS, and GCNVE.

5.5.2 Output Requirements

This program creates at least three tapes from input logical log tape: B1, a miscellaneous data tape; B4, which has high-speed input data on the first file and discrete events data on the second file; and A4, a high-speed output data tape. If necessary, there may be a second high-speed output data tape A5. All redundant and rejected records are placed on the B1 tape for possible future analysis.

5.5.3 Method

SORTER first rewinds the A4, (A5, when necessary) B1 and B4 tapes before writing data onto them. The entry keys are stored and sensed to determine if they contained the number of physical log tapes comprising the logical log tape. If the keys were not set, the program stores an error indication and returns to Monitor.

If the entry keys were set, SORTER initializes to minus one the temporary storage locations for discrete events. All indicators and switches are also set to zero. The program then reads a record from the log tape. Each record consists of ten 17-word blocks. Each block consists of five words of identification and 12 words of data. A test is made to see if the record read is redundant. If a redundancy occurs, the program attempts to read the record ten times. If the redundancy continues to occur, the record is written on the B1 tape and the next record is read.

If the redundancy does not continue or if there was no redundancy, the record is tested for an end-of-file. If the record is not an end-of-file, the log identification for the block is stored in an identification input buffer. The remaining 12 words are stored in a data input buffer. SORTER tests the subchannel number associated with the block. If the subchannel number equals one, the program calls the subroutine GEB. IPORR is called if the subchannel number is two. HSOP1 is called if the subchannel number is three. If the subchannel number is 30, the program calls MANIN.

If SORTER finds that the subchannel number is not 1, 2, 3 or 30, the program determines whether all ten blocks of data have been processed. If the blocks have not been processed, the program continues to the next block of data. If all ten blocks have been processed, the program proceeds to read the next record.

If the record being tested is an end-of-file record, it is determined whether this is the logical end of the log tape. If this is not the logical end of the log tape, the program examines to see if the B6 or B7 was processed. The tape that was processed is rewound and unloaded. All references are changed to the alternate tape (from B6 to B7, or vice versa).

If the end of the logical log tape is reached, SORTER determines whether B6 or B7 was the last tape processed. This last tape is rewound and unloaded. If all tapes have been processed, success is indicated and a search is made for seven discrete events for the program GETME. All data pertaining to these events is written on the second file of the B4 high-speed input data tape.

SORTER finally determines whether the last output record was written on A4 or A5 and writes a double end of file on the appropriate tape. An end of file is also written on B1 and the tapes are rewound and B6 is unloaded.

5.5.4 Usage

Call Statement

CALL SORTER (NOYES)

NOYES is an error indicator

1 indicates error return, error in tape setup

2 indicates normal return

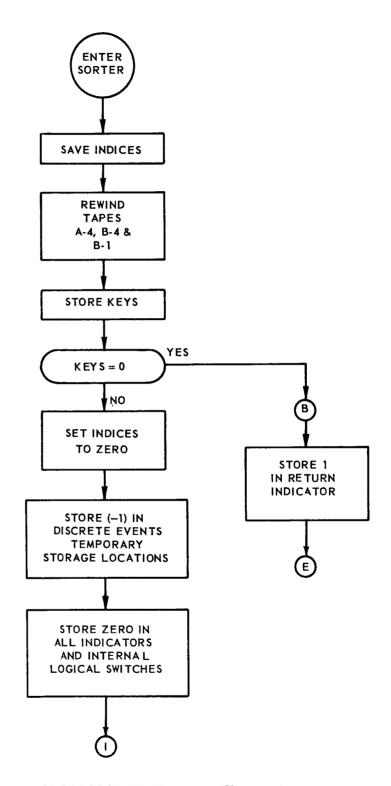


FIGURE 5-4. SORTER PROGRAM FLOW CHART (Sheet 1 of 4)

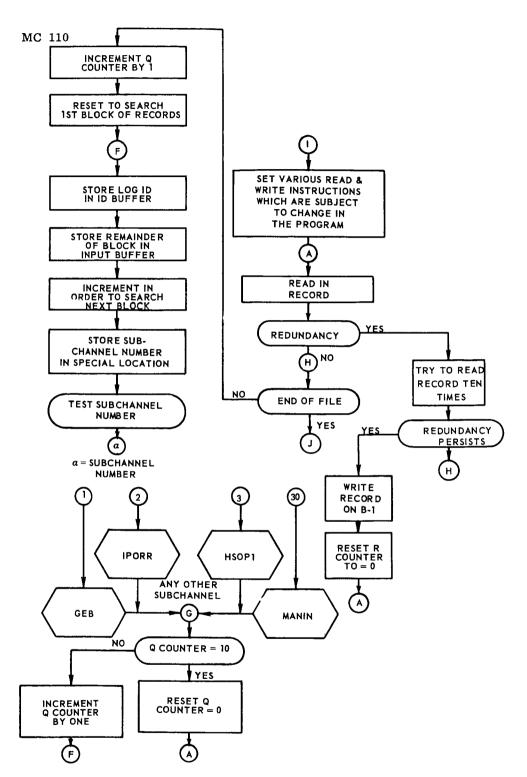


FIGURE 5-4. SORTER PROGRAM FLOW CHART (Sheet 2 of 4)

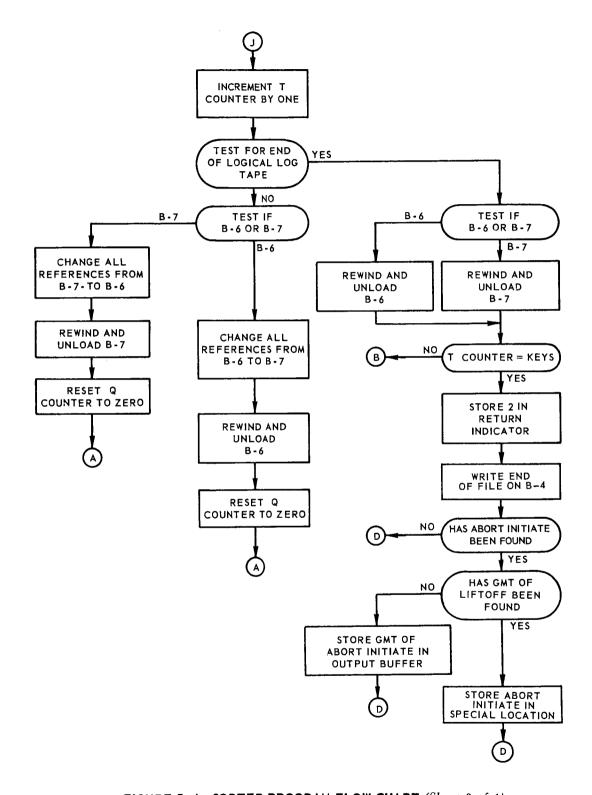


FIGURE 5-4. SORTER PROGRAM FLOW CHART (Sheet 3 of 4)

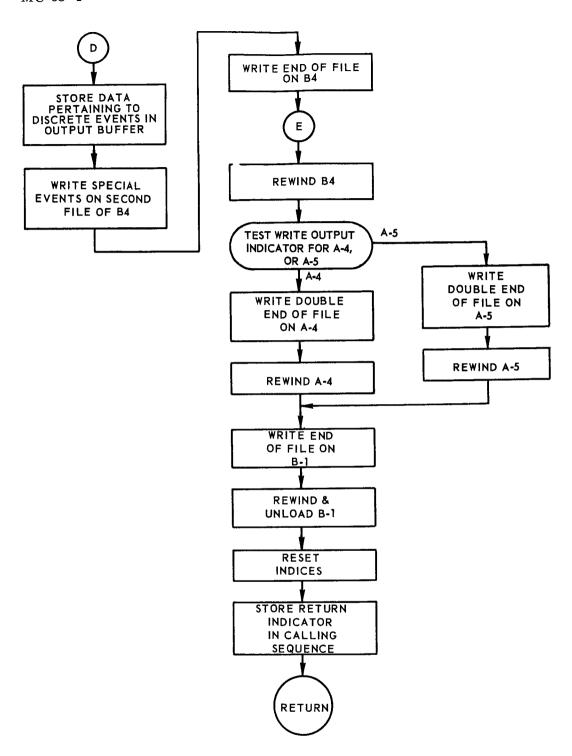


FIGURE 5-4. SORTER PROGRAM FLOW CHART (Sheet 4 of 4)

5.6 SUBCHANNEL 1 PROCESSING PROGRAM (GEB)

GEB processes all subchannel 1 data (high-speed B-GE input from Cape Canaveral). From the output telemetry of this Canaveral data, it determines the occurrence of discrete events. This program also converts position (\vec{r}) and velocity (\vec{v}) vectors to internal units of the Postflight Reporter.

The flow chart for GEB is shown in Figure 5-5.

Using the subroutine I0HSGB, GEB receives a 17-word block to be processed and outputs data on the B1 (miscellaneous) and B4 (high-speed input) tapes. Each complete message consists of four 17-word blocks, and GEB determines which of these is to be processed. If it is the first block of the sequence, the 12 significant data words are stored and the program returns to SORTER to process the next block. If it is the second, third, or fourth 17-word block, a log time test is made.

The log time of each block is compared with that of the first block. If the times are equal, each of 12 significant data words are stored sequentially until a 48-word buffer is filled. The log time is stored in a location used by I0HSGB, and GEB calls I0HSGB to process the 48-word input buffer.

If the log times are not equal, all "ones" (1's) are stored in the remainder of the output buffer. The output buffer is then written on the B1 tape. The log identification is stored in the output buffer and the log time is placed in a location used by I0HSGB.

If IOHSGB finds computed data without error, it exits to MFHSGB. If IOHSGB finds an error in computed data, it exits to MFML6A. Exit is also made to MFML6A in the case of telemetry data, whether good or bad.

When exit is to MFHSGB, the vector time is stored. The \vec{r} and \vec{v} values are converted to feet and feet per second, respectively, and are also stored. If liftoff has occurred, GMT of liftoff is stored and the liftoff indicator is set to nonzero. If liftoff has not occurred, the program processes the next block of data. If SECO has occurred and if the SECO indication did appear, the elapsed time of SECO is stored and the SECO indicator is set to nonzero.

After liftoff is found, the output is written on B4; if correct, and it is determined whether this is the end of the tape. If it is the end of B4, an end of file is written and B4 is rewound and unloaded. Whether or not an end of tape appears, GEB returns to SORTER to process the next block of data.

When exit is to MFML6A, it is determined if an error occurred in the data and, if so, whether the error was due to the previously mentioned time test. If the error was a result of the time test, counters, indicators, and addresses are reset and GEB restarts processing. If the error did not result from the

time test, the data is read into an error buffer, error is indicated and written on the B1 tape, and GEB returns to SORTER.

If an error did not occur, the data is presumed to be telemetry data, and the time tags, discrete signals and B-GE selected source, are stored. GEB then tests for liftoff, abort initiate, retrorocket firings, abort phase inception, and orbit phase inception.

If liftoff occurred in the message being examined, then GMT of liftoff is stored and the liftoff indicator is set. If neither abort initiate nor SECO has occurred previously but abort initiate occurs in the present message, GMT of abort initiate is stored and the SECO and abort initiate indicators are set.

The program then tests whether the first, second, or third of three retrorockets have already been fired. If the first or second or none of the retrorockets have been previously fired, GEB tests the present message for an indication of retrorocket firing. If the retrorockets did fire, the time of retrofire and the number of retrorockets fired are stored.

If the retrorockets did not fire or if the firing of the third retrorocket was previously indicated, the program tests for abort and orbit phase inception. If either occurred previously, or if either cannot be found in a previous or in the present message, GEB returns to SORTER. If either is found in the message under examination, the appropriate indicator is set to nonzero, the GMT's of inception of the events are stored, and the program returns to SORTER.

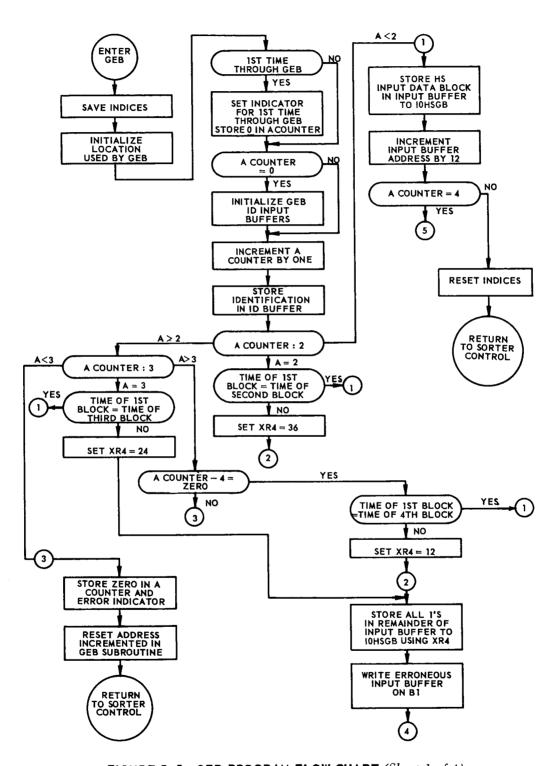
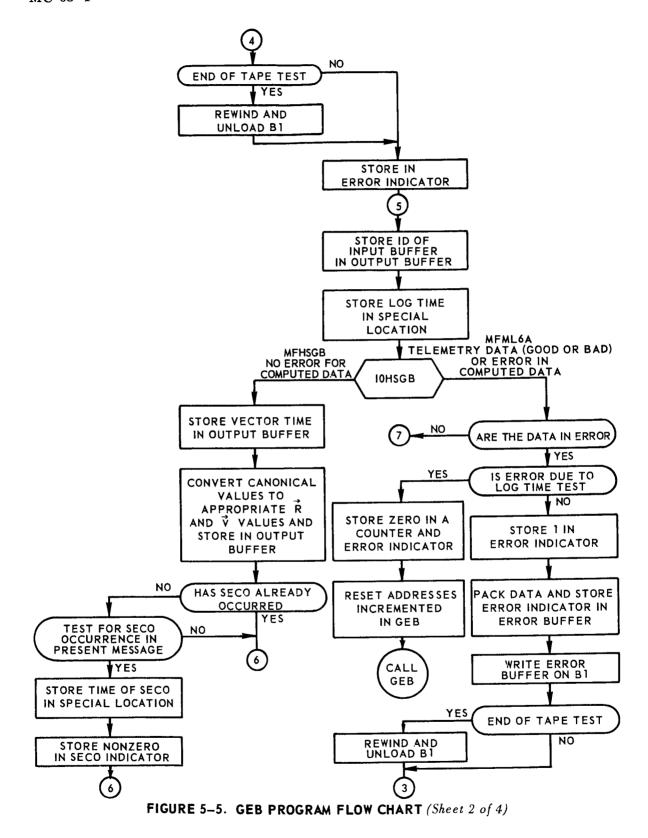


FIGURE 5-5. GEB PROGRAM FLOW CHART (Sheet 1 of 4)



5-30

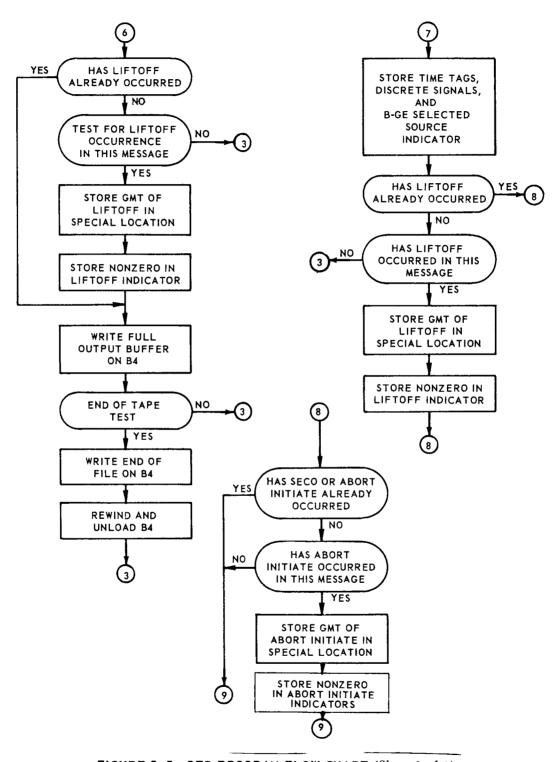


FIGURE 5-5. GEB PROGRAM FLOW CHART (Sheet 3 of 4)

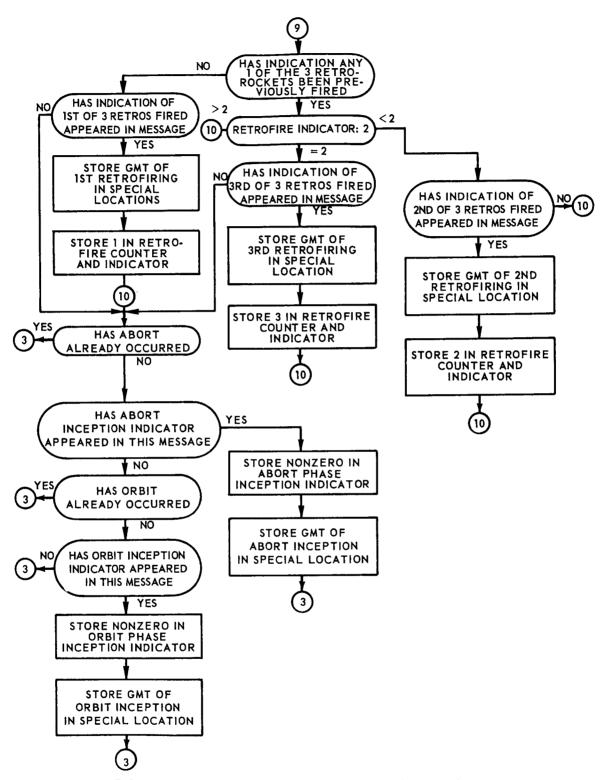


FIGURE 5-5. GEB PROGRAM FLOW CHART (Sheet 4 of 4)

5.7 IP 7094 HIGH-SPEED INPUT PROCESSOR PROGRAM (IPORR)

IPORR processes all high-speed input data from the Cape Canaveral IP 7094 computer. This program receives a 17-word block of data from an input buffer. Using the subroutine I0HS09 (see Volume MC 63-3, <u>Goddard Processor Programs</u>), IPORR processes the IP 7094 high-speed input and places significant data on the high-speed input tape (B4). The erroneous data is packed and stored in an error buffer and written on the miscellaneous tape (B1).

The flow chart for IPORR is shown in Figure 5-6.

When IPORR is entered and a block of data is input, the identification of the block is stored in an identification buffer and the program determines whether this is the first, second, third or fourth block of a sequence. If it is the first block, the 12 significant data words are stored in the IOHS09 input buffer. If it is the second, third or fourth block, a time test is made.

The time associated with each sequential block is compared with the time of the first block. If the times agree, the data in the block being compared with the first block is stored sequentially until a 48-word buffer is filled. If the times do not agree, all "ones" (1's) are stored in the remainder of the buffer and written on the B1 tape. An error indicator is also set. Whether or not the times are equal, the identification of the block is stored in an output buffer, and the log time is stored in a location used by I0HS09, which is then called.

I0HS09 exits to MFHS09 if the data source is IP 7094 processed data. If the data source is the IP 7094, the position (\vec{r}) and velocity (\vec{v}) vectors are converted and stored with the respective time tag in the output buffer. The program tests for liftoff at this time. If liftoff did not already occur, the program returns to process the next block. If liftoff did occur, the output buffer is written on the B4 tape and the program returns to process the next block.

IOHS09 exits to MFHS08 if the data is assumed to be either telemetry or erroneous. If the data is erroneous and the error is a result of the previously mentioned time test, the program returns to IPORR. If the error is not from the time test, the data is packed, stored, and written on B1, and the program then returns to SORTER.

If no error is indicated, the program stores time tags, discrete messages, and IP selected source indicator. The program then tests for liftoff. If it has not occurred previously, but does occur in the message being examined, GMTLO is stored and the liftoff indicator is set to nonzero.

IPORR also tests for abort initiate, first, second and third retrofire, abort phase inception, and orbit phase inception. If each of these is found, the GMT of its occurrence is stored and the appropriate indicator is set to nonzero. Whether or not the discrete events occurred, the program returns to SORTER to process the next block of data.

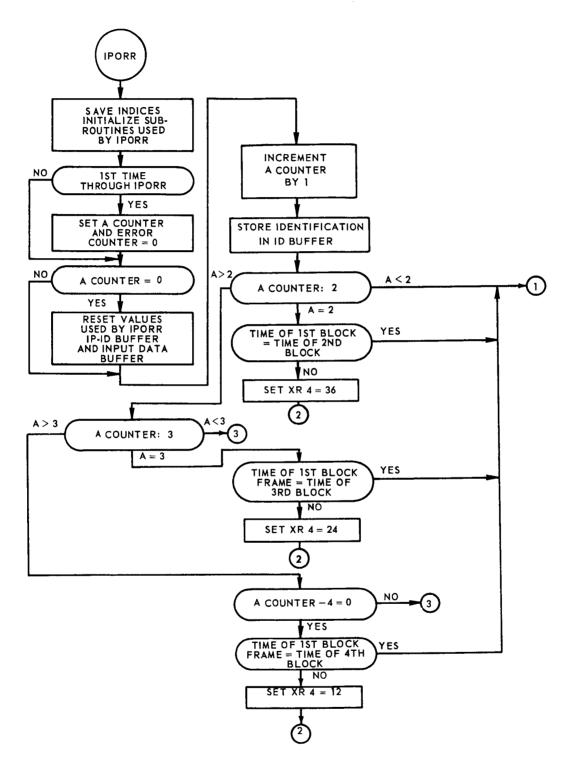


FIGURE 5-6. IPORR PROGRAM FLOW CHART (Sheet 1 of 5)

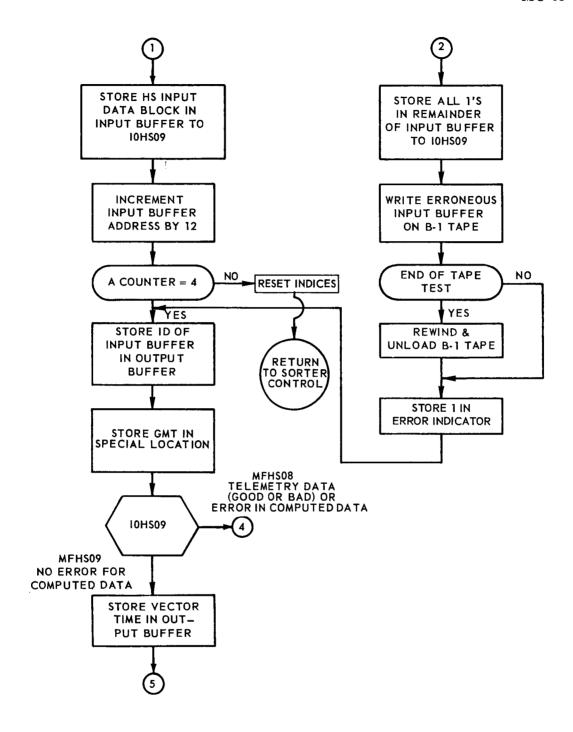


FIGURE 5-6. IPORR PROGRAM FLOW CHART (Sheet 2 of 5)

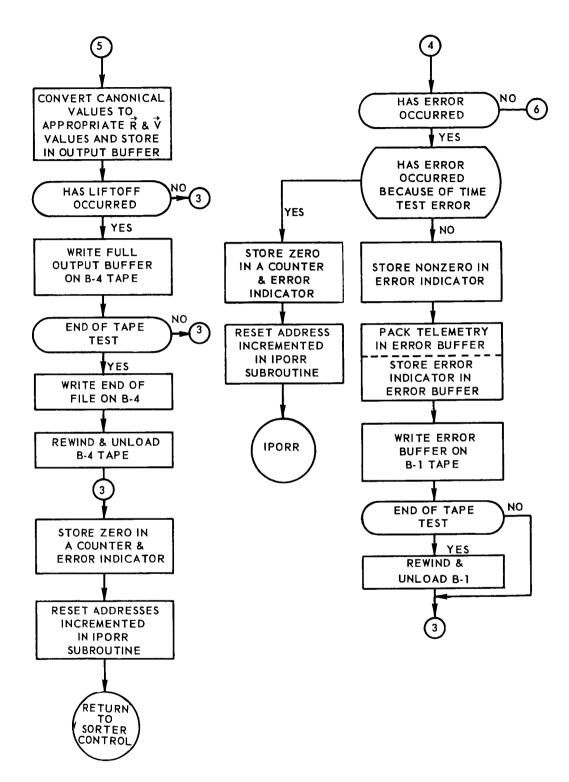


FIGURE 5-6. IPORR PROGRAM FLOW CHART (Sheet 3 of 5)

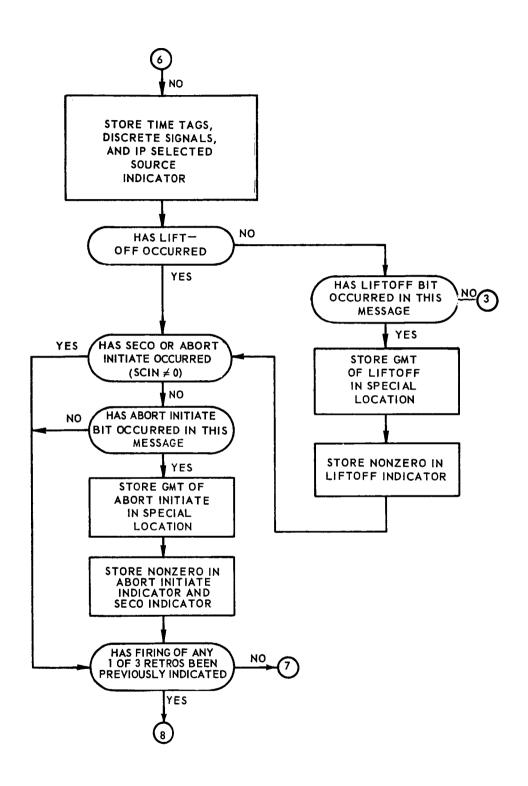


FIGURE 5-6. IPORR PROGRAM FLOW CHART (Sheet 4 of 5)

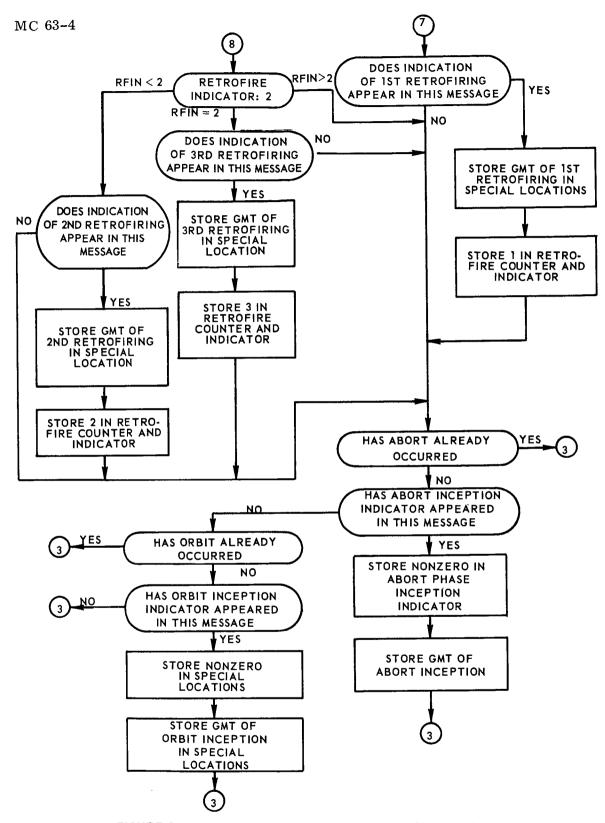


FIGURE 5-6. IPORR PROGRAM FLOW CHART (Sheet 5 of 5)

5.8 MANUAL INSERTION PROCESSOR PROGRAM (MANIN)

MANIN searches manually inserted messages for the occurrence of discrete events. This program uses IOMANI as a subroutine.

The flow chart for MANIN is shown in Figure 5-7.

The program is entered and a word count is taken from the block identification and stored in the decrement of a location already containing the address of the first word of an input buffer. The data is stored and the location is placed into the AC. MANIN calls IOMANI to process the data. When IOMANI exits to MFMANI, MFMAN3, MFMAN5, MFMAN6, MFMAN8, or MFMAN9, MANIN returns to SORTER to process the next block of data.

If liftoff data were in the message, IOMANI exits to MFMAN1. Thereupon, MANIN converts GMT of liftoff (GMTLO) to floating-point, stores it, and sets the liftoff indicator equal to nonzero. This GMTLO overrides any previous GMTLO found by any other subroutine. The program returns to SORTER to process the next block of data.

An exit to MFMAN2 indicates retrofire data. The number of retrorockets fired is stored. A three (3) is placed in the retrofire indicator, and the GMT of the first retrofire is converted to floating-point and stored. This overrides any previous retrofire time. Again the program returns to SORTER.

When IOMANI exits to MFMAN4, an incremental delta t for either the IP 7094 or B-GE vectors is to be added. In this case, the delta t for both types of vectors is converted to floating-point and stored. MANIN returns to SORTER.

If exit is made to MFMAN7, the nonnominal retrofire data is processed. The GMT of the first retrorocket firing is stored in floating-point in the output along with the number of retros fired. MANIN returns to SORTER.

Exit is made to MFMAOS on indication of abort or orbit data. A test is made to see if abort phase inception has been indicated. If not, checks are made for indication of orbit inception in the message. If orbit inception does not appear in the present message, the program returns to SORTER to process the next block of data. If the present message indicates that either orbit or abort phase is in effect, the respective indicators are set to nonzero, the time of the message is placed in floating-point mode and stored, and the program returns to SORTER.

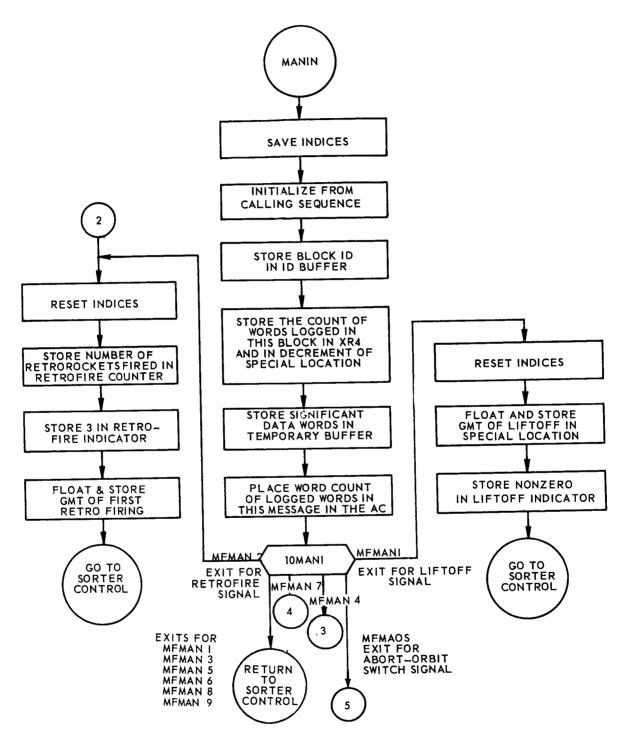


FIGURE 5-7. MANIN PROGRAM FLOW CHART (Sheet 1 of 2)

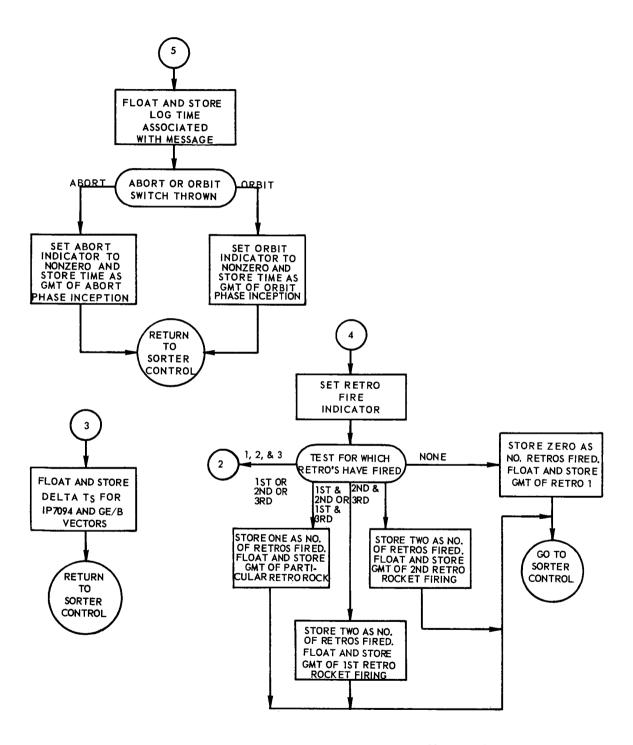


FIGURE 5-7. MANIN PROGRAM FLOW CHART (Sheet 2 of 2)

5.9 HIGH-SPEED OUTPUT TAPE WRITER PROGRAM (HSOP1)

This program processes all high-speed output data and writes it on the A4 tape (and A5 if necessary). HSOP1 converts and scales the data with the use of five conversion subroutines.

The flow chart for HSOP1 is shown in Figure 5-8.

Input to this program are two 17-word data blocks. HSOP1 also uses the conversion subroutines BCTB (BCTB1), BCTBI (BCTBJ), HMSTS, TISWS, and GCNVE. The program outputs a 44-word buffer and an A4 tape.

HSOP1 tests whether the occurrence of liftoff has been indicated. If it has not, the program returns to SORTER to process the next block of data. If lift-off has been indicated, a test is made to determine whether the first or second subframe is being processed. In either case, the ID buffer is set up for the appropriate ID. Then the proper subframe ID and the phase indication are stored in appropriate locations. Again a subframe test is made. Now, if the first subframe is being examined, the eight significant data words are stored in an intermediate input buffer. A test is then made for the correct phase. If the test proves that the phase indication is not recognizable, the erroneous block of data is written on B1 and the routine returns to SORTER to process the next block of data. Yet if the phase indication is acceptable, the correct phase is stored and the correct odd-even indication is also stored. Then the routine returns to SORTER.

If the second subframe is being examined, this means a complete message has been received and the five significant words are stored in the remaining portion of the intermediate input buffer. A test is made for indication of reentry phase in this message or a previous message. If this is the first indication of reentry phase inception, the time of the message is stored in floating-point mode as the GMT of reentry phase. In any case, the routine then proceeds to unpack all strip charts and wall map data and store the data with the data source indicator in the output buffer. A test is then made to see if the message is odd or even frame data. If the message is an odd frame, the digital displays are masked. GTRS and GMTLC are converted by HMSTS: $\varphi_{\rm IP}$ is converted by BCTB; and γ (or HA) is converted by BCTBJ. All converted values are stored in the output buffer.

If the message is an even frame, the digital displays are masked and converted. GMTRC $_s$, ECTRC $_s$, Δt_r , and GMTRS are converted by HMSTS; ICTRC is converted by TISWS. All converted values are stored in the output buffer.

Whether the message frame is odd or even, all the plotboard data is extracted and stored in the output buffer. All the nondigital display data is converted by GCNVE to standard values and replaced in the same output buffer. All data in the output buffer is put in floating-point mode. This output buffer is written on A4 (or A5) together with the discrete signals.

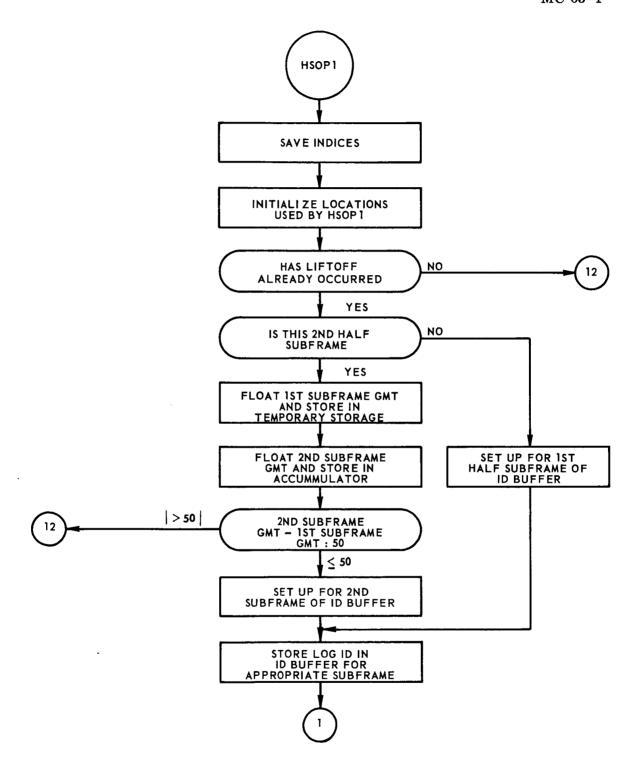
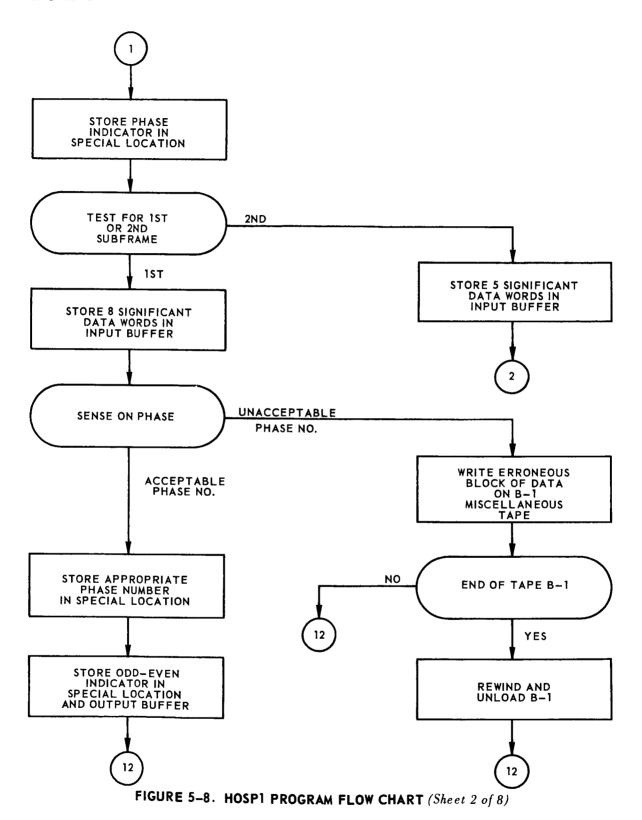


FIGURE 5-8. HOSP1 PROGRAM FLOW CHART (Sheet 1 of 8)



5-46

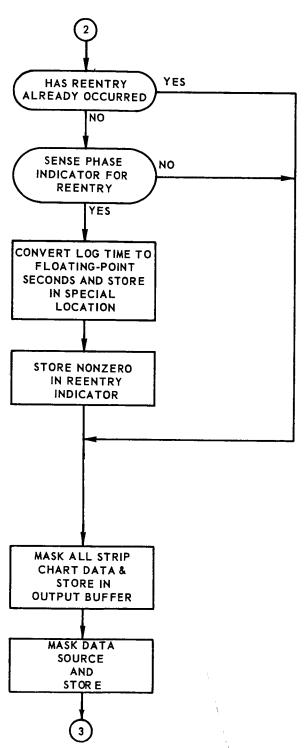


FIGURE 5-8. HOSP1 PROGRAM FLOW CHART (Sheet 3 of 8)

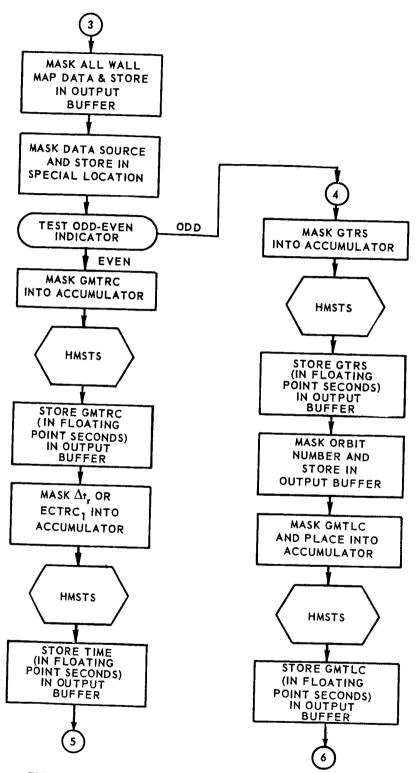


FIGURE 5-8. HOSPI PROGRAM FLOW CHART (Sheet 4 of 8)

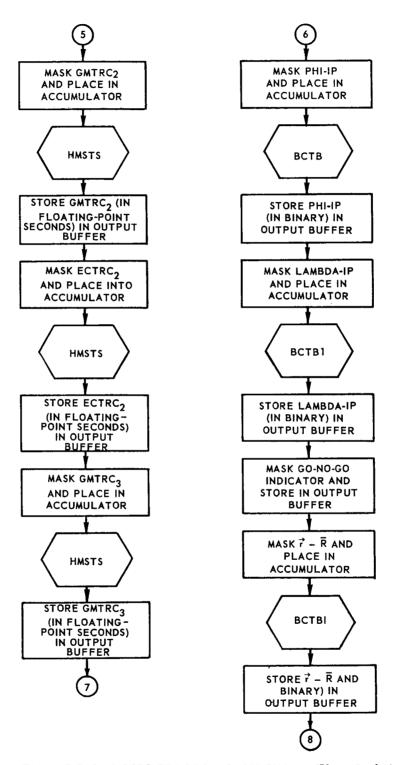


FIGURE 5-8. HOSP1 PROGRAM FLOW CHART (Sheet 5 of 8)

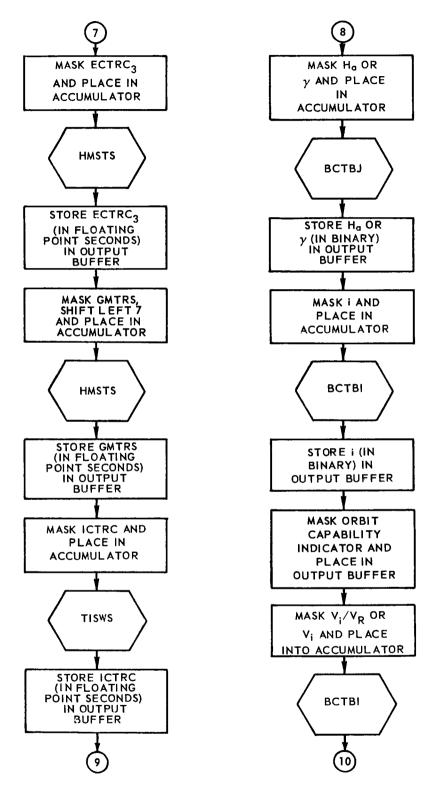


FIGURE 5-8. HOSP1 PROGRAM FLOW CHART (Sheet 6 of 8)

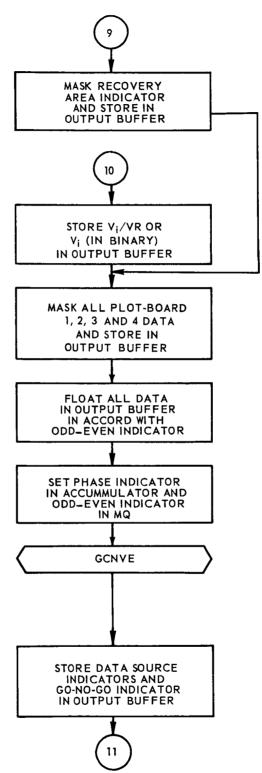


FIGURE 5-8. HOSP1 PROGRAM FLOW CHART (Sheet 7 of 8)

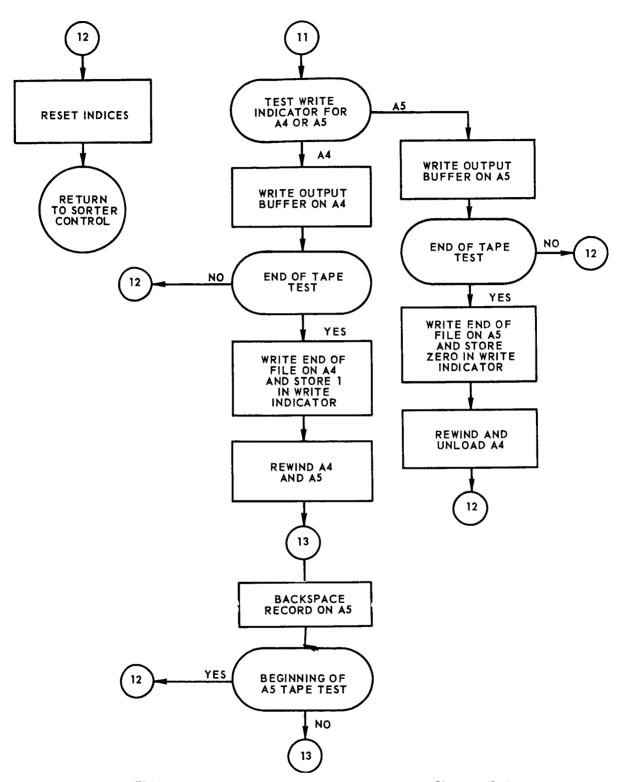


FIGURE 5-8. HOSP1 PROGRAM FLOW CHART (Sheet 8 of 8)

5.10 BCD WORD CONVERSION PROGRAM (A) (BCTB/BCTB1)

This subroutine of HSOP1 converts modified BCD numbers to fixed-point binary numbers. The modified BCD numbers are in one of two prescribed formats, which are prerequisites for proper entry into the subroutine.

The flow chart for BCTB/BCTB1 is shown in Figure 5-9.

This program is entered through BCTB if the format of the modified BCD number is 4 bits/digit, 4 bits/digit, 3 bits/digit and 4 bits/digit. Entry is through BCTB1 if the format of the word to be converted is 1 bit/digit, 4 bits/digit, 4 bits/digit, 3 bits/digit and 4 bits/digit.

The modified BCD number should be in the AC at time of entry. The converted binary digits are summed and placed into the AC upon exit.

The calling sequences for both entries are:

alpha TSX \$BCTB (or \$BCTB1)

+1 Normal return

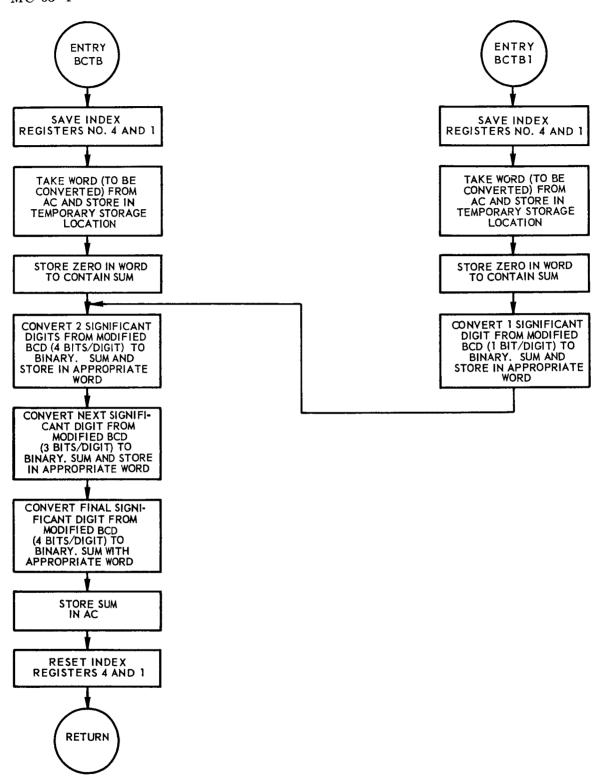


FIGURE 5-9. BCTB/BCTB1 PROGRAM FLOW CHART

5.11 BCD WORD CONVERSION PROGRAM (B) (BCTBI/BCTBJ)

This subroutine of HSOP1 converts modified BCD numbers to fixed-point binary numbers. The modified BCD numbers are in one of two prescribed formats, which are prerequisites for proper entry into the subroutine.

The flow chart for BCTBI/BCTBJ is shown in Figure 5-10.

The modified BCD numbers are in 4 bits/digit format. If the input number is not preceded by a sign, the subroutine is entered through BCTBI. If the number is preceded by a sign, the subroutine is entered through BCTBJ. In this latter case, the appropriate sign is set in the AC. However, both subroutines convert, sum the digits, and place the fixed-point binary numbers into the AC upon exit.

The calling sequences for both entries are:

alpha TSX \$BCTB1 (or \$BCTBJ)

+1 Normal return

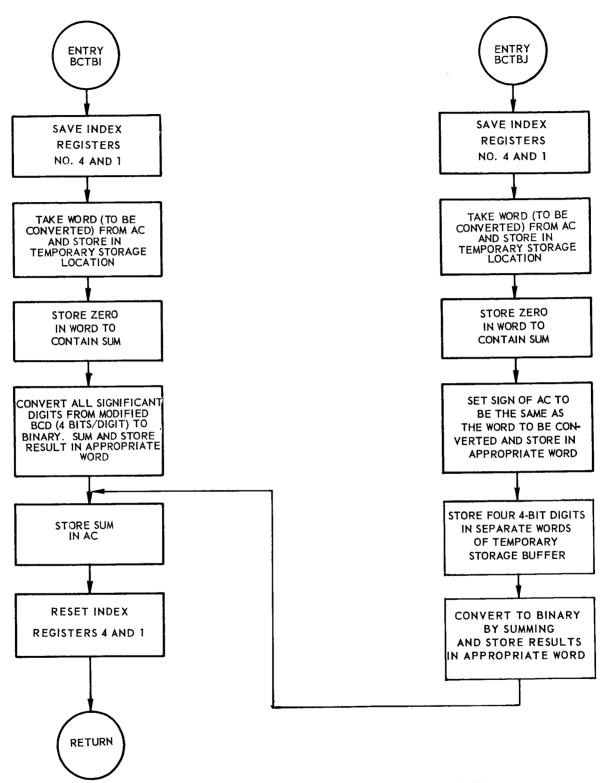


FIGURE 5-10. BCTBI/BCTBJ PROGRAM FLOW CHART

5.12 TIME WORD CONVERSION PROGRAM (A) (TISWS)

This subroutine of HSOP1 converts a modified BCD number representing a time (in hours, minutes and seconds) to a binary number representing the same time (in fixed-point seconds).

The flow chart for TISWS is shown in Figure 5-11.

The modified BCD number, placed in the AC when the subroutine is entered, is given in the following format:

1 bit - sign for hours
2 bits/digit - tens of hours
4 bits/digit - unit of hours
1 bit - sign for minutes
3 bits/digit - tens of minutes
4 bits/digit - unit minutes
1 bit - sign for seconds
3 bits/digit - tens of seconds
4 bits/digit - unit seconds

Each digit is converted and/or scaled to its appropriate equivalent in fixed-point seconds with the correct sign. The binary equivalents (with sign) are summed and stored in the AC when the program exits.

The calling sequence is:

alpha TSX STISWS

+1 Normal return

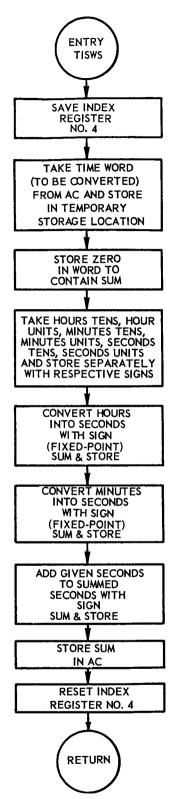


FIGURE 5-11. TISWS PROGRAM FLOW CHART

5.13 TIME WORD CONVERSION PROGRAM (B) (HMSTS)

This subroutine of HSOP1 converts a modified BCD number, representing a time, to a binary number representing the same time (in fixed-point seconds).

The flow chart for HMSTS is shown in Figure 5-12.

The modified BCD number, placed in the AC when the subroutine is entered, is shown in one of the following two formats:

Format 2

2 bits/digit - tens of hours	2 bits/digit - tens of hours
4 bits/digit - unit hours	4 bits/digit - unit hours
3 bits/digit - tens of minutes	3 bits/digit - tens of minutes
4 bits/digit - unit minutes	4 bits/digit - unit minutes
	3 bits/digit - tens of seconds
	4 hits/digit - unit seconds

Each digit is converted and/or scaled to its appropriate equivalent in fixed-point seconds. The binary equivalents are summed and stored in the AC when the program exits.

The calling sequence is:

Format 1

alpha	TSX	\$HMSTS
+1		Normal return

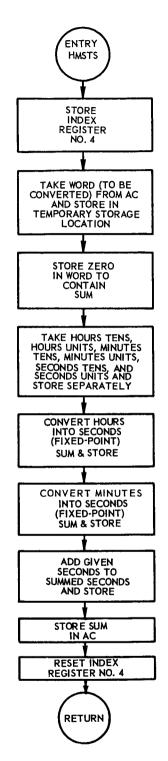


FIGURE 5-12. HMSTS PROGRAM FLOW CHART

5.14 UNIT CONVERSION PROGRAM (GCNVE)

This subroutine is used by HSOP1 to convert and scale high-speed output from granular values to significant standard values. The high-speed output may be from the strip chart (launch phase only), plotboard 3 (orbit phase only), and/or from the wall map and plotboards 1, 2 and 4 (all phases).

The flow chart for GCNVE is shown in Figure 5-13.

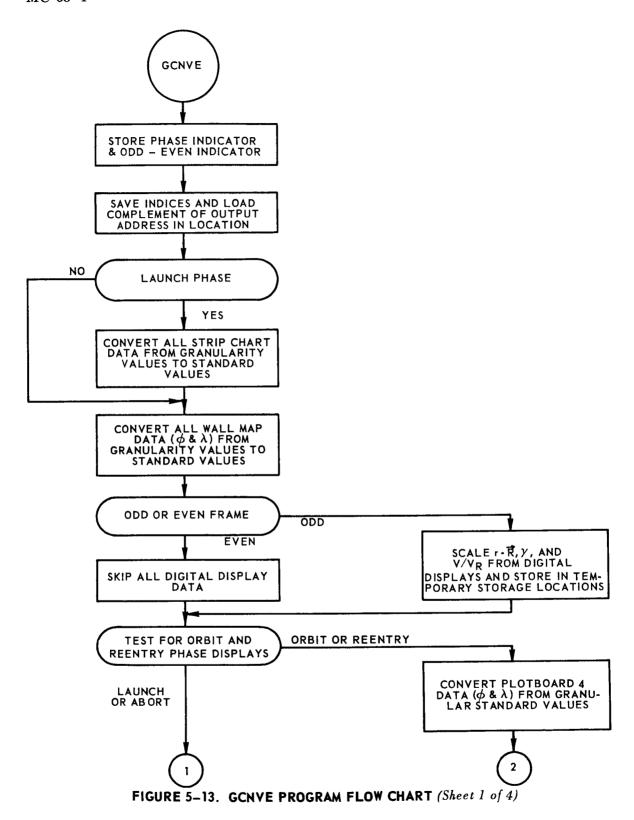
The address of the first location in the buffer to be converted is given in the PZE of the calling sequence. The buffer contains all high-speed output data for conversion, then the program is entered through GCNVE. The AC contains a phase indicator and the MQ contains an odd-even indicator.

The program stores the phase and odd-even indicator from the AC and MQ, saves the indices and stores the complement of the first word of the address of the buffer. The program then converts all the aforementioned data according to phase, odd-even indication, and limits set up on parameters (D and H of plot-board 2 in launch and abort phases and V/VR of plotboard 1 in launch phase).

The buffer area now contains all converted and scaled high-speed output data when the indices are reset and the program returns control to HSOP1.

The calling sequence is:

alpha	TSX	\$GCNVE
+1	PZE	(address of first location in buffer to be converted)
+ 2		Normal return



5-62

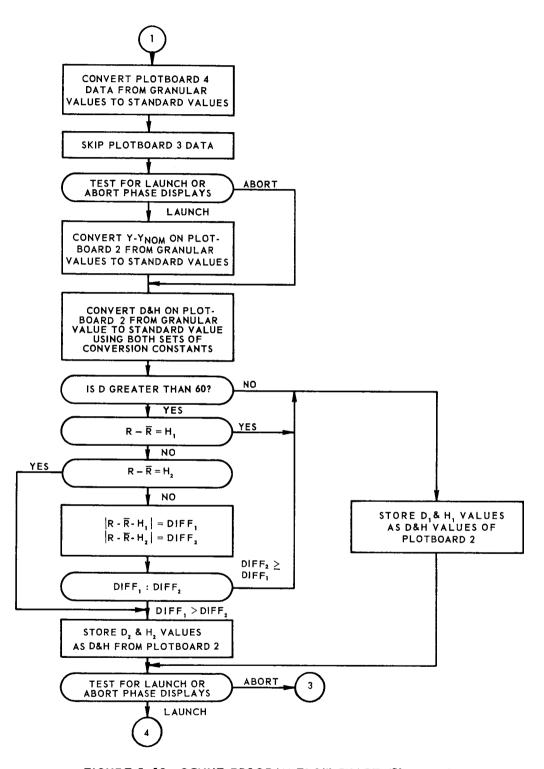


FIGURE 5-13. GCNVE PROGRAM FLOW CHART (Sheet 2 of 4)

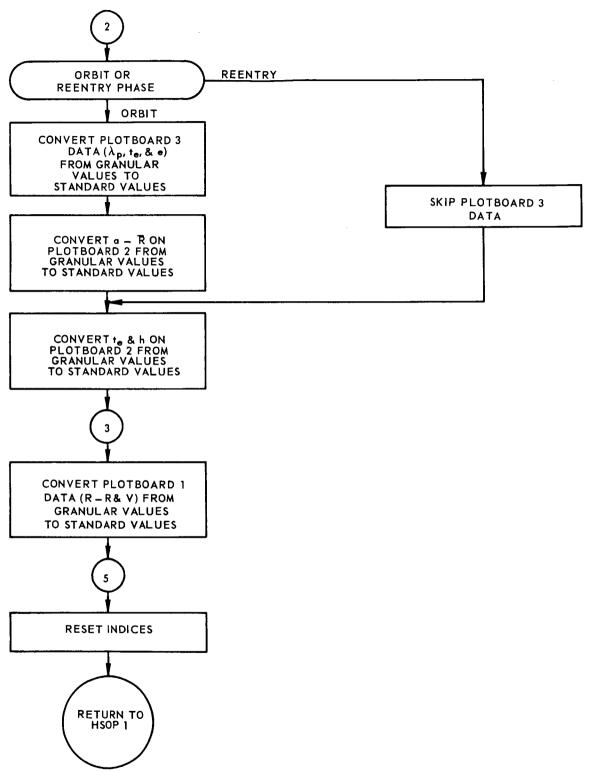


FIGURE 5-13. GCNVE PROGRAM FLOW CHART (Sheet 3 of 4)

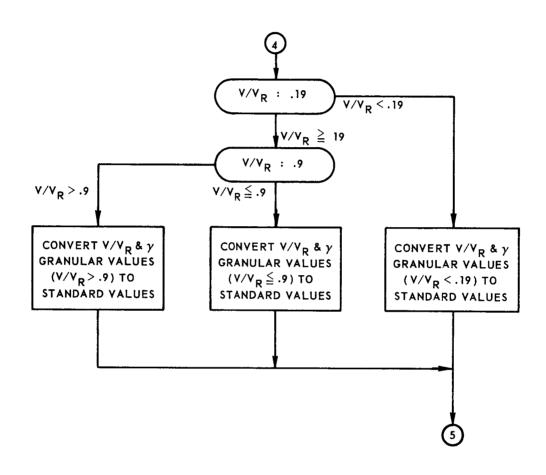


FIGURE 5–13. GCNVE PROGRAM FLOW CHART (Sheet 4 of 4)

5.15 DISCRETE EVENT PROCESSOR PROGRAM (GETME)

GETME searches the second file of the high-speed input tape (B4) created by SORTER from data on the Mercury Programming System log tape. This second file contains a record of the occurrence of seven discrete events. The information located on these events is placed in COMMON storage for reference by Monitor. However, the Postflight Reporter Program cannot continue beyond the GETME program if the first word of this second file, the GMT of liftoff, is not found.

The flow chart for GETME is shown in Figure 5-14.

5.15.1 Input Requirements

The input to GETME is the second file of the B4 tape created by the SORTER program. This file contains a record of all data pertaining to the occurrence of seven discrete events.

5.15.2 Output Requirements

GETME records the following information about the discrete events:

- a) Greenwich Mean Time of liftoff (GMTLO).
- b) Number of retrorockets fired.
- c) Elapsed time from liftoff to sustainer engine cutoff (SECO).
- d) Time of abort inception referenced to liftoff.
- e) Time of orbit inception referenced to liftoff.
- f) Time of reentry inception referenced to liftoff.
- g) Time of firing first retrorocket referenced to liftoff.

This information is placed into COMMON storage before the exit of this program.

5.15.3 Method

This program skips the first file on the high-speed input tape (B4) and reads the second file for the data pertaining to the discrete events. If the record is not accepted, the program places a one (1) in the error indicator. Another attempt may be made at obtaining the data.

When the record is accepted, a test is made for GMT of liftoff. If GMT of liftoff is not found, the B4 tape is rewound, and the program returns to Monitor with a one (1) in the error indicator. If GMT of liftoff is found, GETME then tests for the existence of the other events within the record. If an event did not occur, the particular location for that event is supplied with a minus one (-1). All data is then transferred to COMMON storage, and all times are referenced to GMT of liftoff (GMTLO).

5.15.4 Usage: Call Statement

CALL GETME (NOYES)

NOYES is an error indicator:

1 indicates an error in tape setup

2 indicates normal return

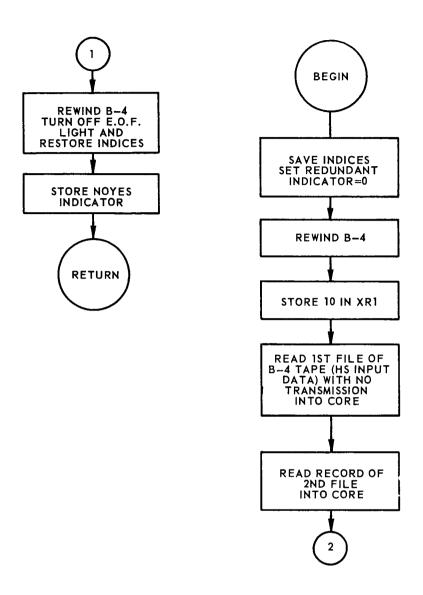


FIGURE 5-14. GETME PROGRAM FLOW CHART (Sheet 1 of 2)

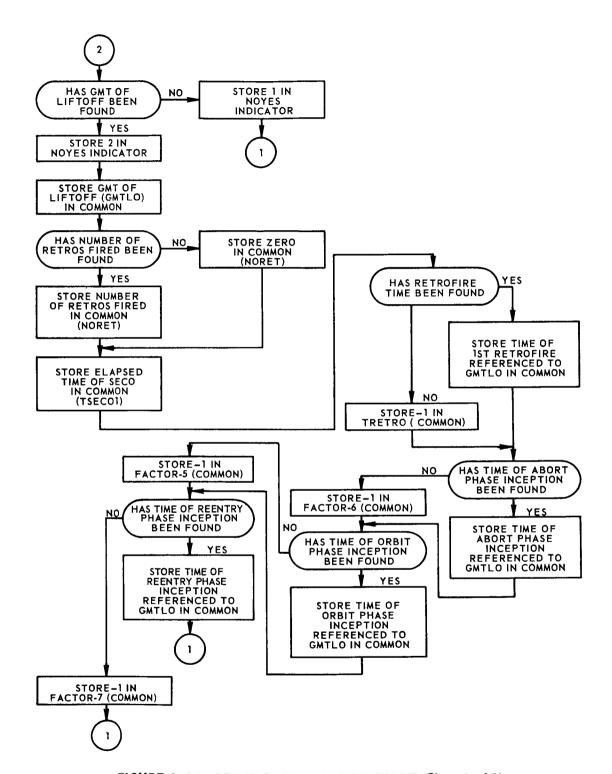


FIGURE 5-14. GETME PROGRAM FLOW CHART (Sheet 2 of 2)

5.16 HIGH-SPEED OUTPUT PROCESSOR PROGRAM (DONOUT)

DONOUT processes the high-speed output tape (A4 and/or A5), which contains the data removed from the Mercury Programming System log tape and unpacked by SORTER. DONOUT removes the data from the A4 (or A5) tape according to a predetermined time and flight phase. It places available data into locations of COMMON storage corresponding to the specific high-speed output quantities.

The flow chart for DONOUT is shown in Figure 5-15.

5.16.1 Input Requirements

The input to this program is high-speed output data written on the A4 tape (and A5 tape if enough room is not available on a single tape). The data is in 44-word records. These records consist of 10 words of heading and 34 words of decoded data. At the physical end of an A4 (or A5) tape is a single end of file. At the logical end of all tapes are two ends of file.

A closed subroutine called BRINR is also used within DONOUT.

5.16.2 Output Requirements

DONOUT places into COMMON storage the high-speed output messages that drove the displays at Cape Canaveral. These messages are obtained according to a unique time tag. In the case of processed information, the time tag is the vector time associated with the input message which gave rise to the output. If such a vector time exists, the data source and vector time are output. If the vector time does not exist, an error return is indicated. Then, if no processed information is available, log time is used instead of vector time.

5.16.3 Method

DONOUT reads a high-speed output record from the A4 (or A5) tape. The phase (launch, abort, orbit or reentry) is determined and compared with the phase requested in the call statement (see below). If the phase of the data is numerically less than the phase requested, the phase requested is further down the tape and another record is read. If the phase of the data is greater than the phase requested and if the requested phase is different from the one previously requested, then the A4 (or A5) tape is rewound so that a search may be made for the new phase.

If the phase of the high-speed output block is the same as the one requested, the time of the output message is examined. If the time of the message is less

than the time requested, the next record is read. If the time of the message is greater than the time requested, the program backspaces until a message of the correct time and phase is determined. If the time requested for the particular phase does not exist within the phase, the time closest (but earliest) to the requested time is accepted. If this phase time is not within a reasonable tolerance, an error is indicated.

Should the tape contain data from one phase, receive data from a second phase and subsequently have more data from the original phase, the program backspaces the tape to determine whether there have been two changes of phase. If possible, DONOUT will locate the last message in the proper phase and present it for decoding.

When records are read from the A4 (or A5) tape, tests are made for redundancies and ends of file. To read the next record, a closed subroutine called BRINR is used within DONOUT. This routine reads in the next record, examines for redundancy and end of file, and then exits. If redundancies exist, backspacing action is taken to reread these records. If the records can be reread, they are used; if they cannot be reread, the tape spaces ahead to the next record.

If an end of file on A4 (or A5) exists, the tape is read a second time to determine whether a second end of file is on it. If so, this is the end of the data and an exit is made. If only one end of file exists on the tape, the physical end of the tape is reached and the alternate tape should be used. The program has the capacity to switch back and forth between A4 and A5 until enough tapes have been read to process all the data.

5.16.4 Usage: Call Statement

CALL DONOUT (TEMPO1, J23, JERROR, NDATA)

TEMPO1 is a time indicator containing the requested vector time.

J23 is a phase indicator containing one of the following:

- 1 indicates launch phase (MA or MR)
- 2 indicates abort phase (MR)
- 3 indicates orbit phase (MA)
- 4 indicates reentry phase (MA)
- 5 indicates high or medium abort phase (MA)
- 6 indicates low abort phase (MA)

JERROR is an error indicator containing one of the following:

- 1 indicates no more output on the log tape for the requested phase
- 2 indicates normal return and normal transmission
- 3 indicates redundancy

NDATA is a data source indicator in which:

- 1 indicates IP 7094 data
- 2 indicates B-GE data
- 3 indicates raw radar data

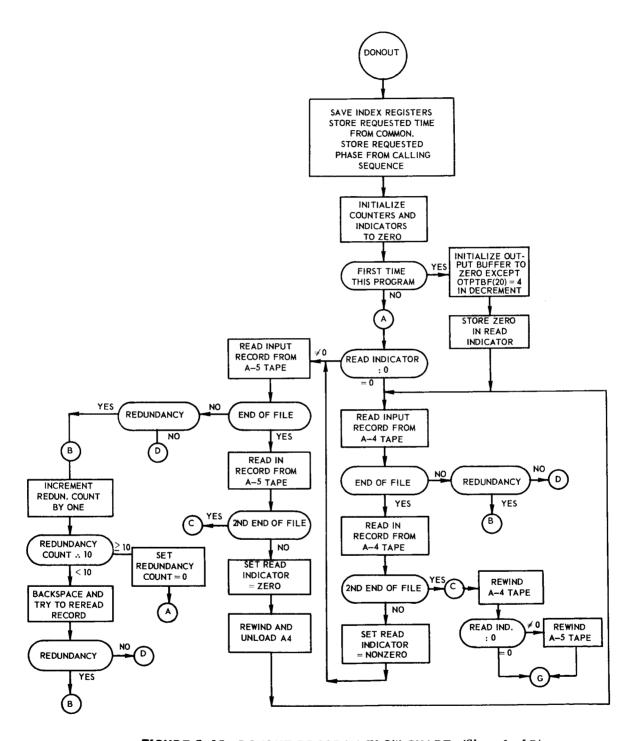


FIGURE 5-15. DONOUT PROGRAM FLOW CHART (Sheet 1 of 7)

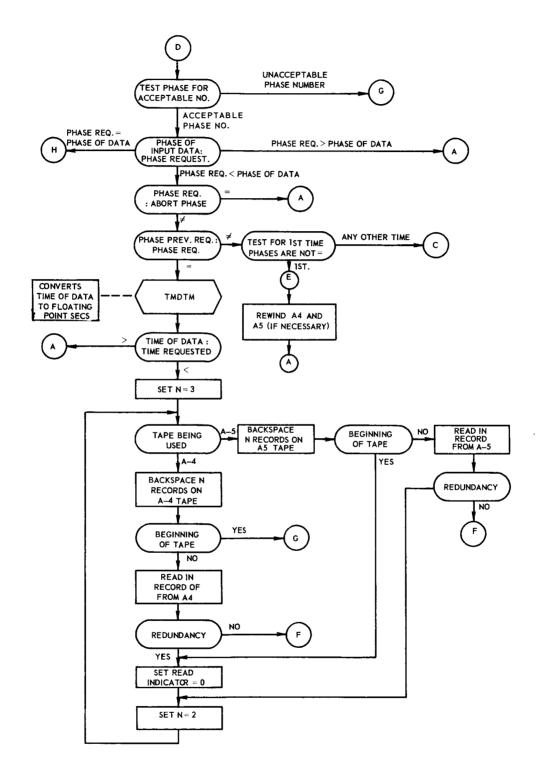


FIGURE 5-15. DONOUT PROGRAM FLOW CHART (Sheet 2 of 7)

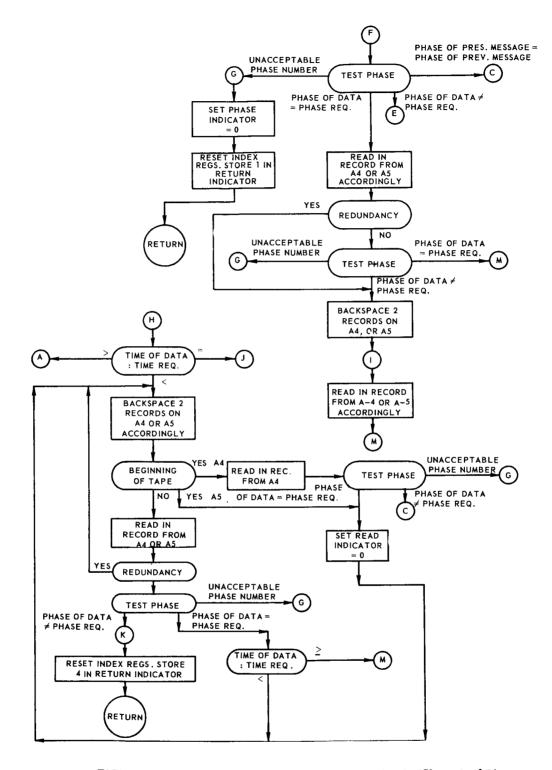


FIGURE 5-15. DONOUT PROGRAM FLOW CHART (Sheet 3 of 7)

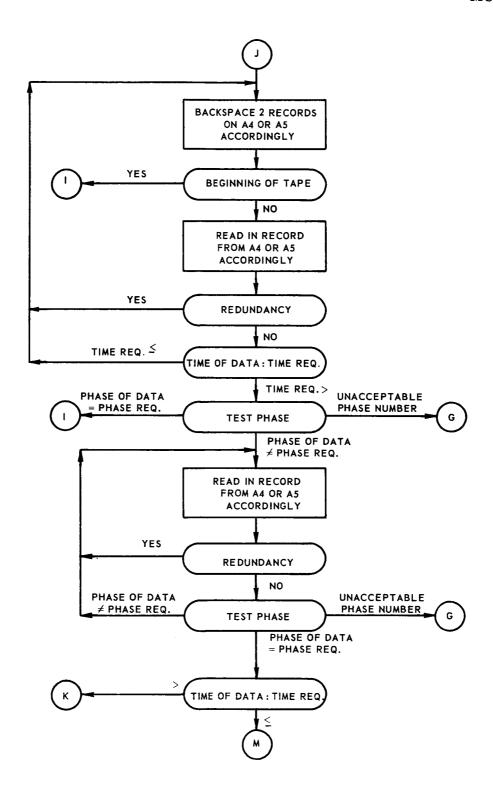


FIGURE 5-15. DONOUT PROGRAM FLOW CHART (Sheet 4 of 7)

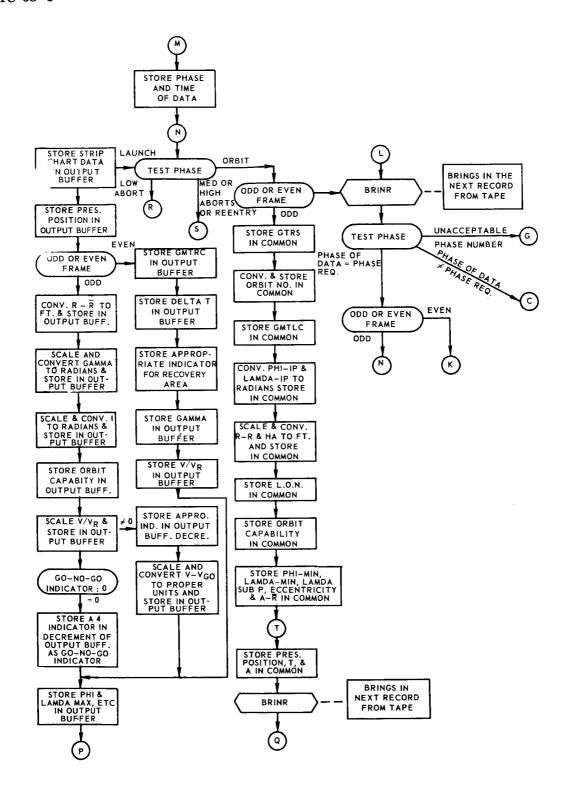


FIGURE 5-15. DONOUT PROGRAM FLOW CHART (Sheet 5 of 7)

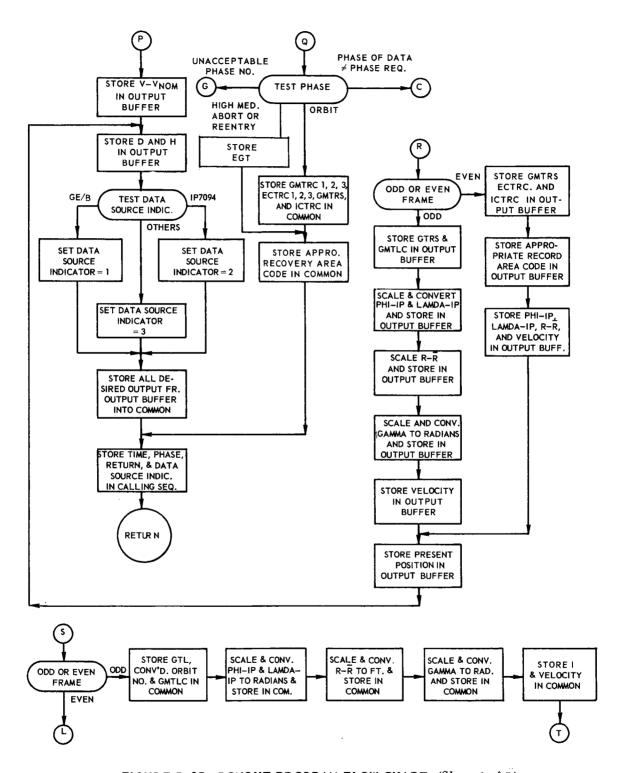


FIGURE 5-15. DONOUT PROGRAM FLOW CHART (Sheet 6 of 7)

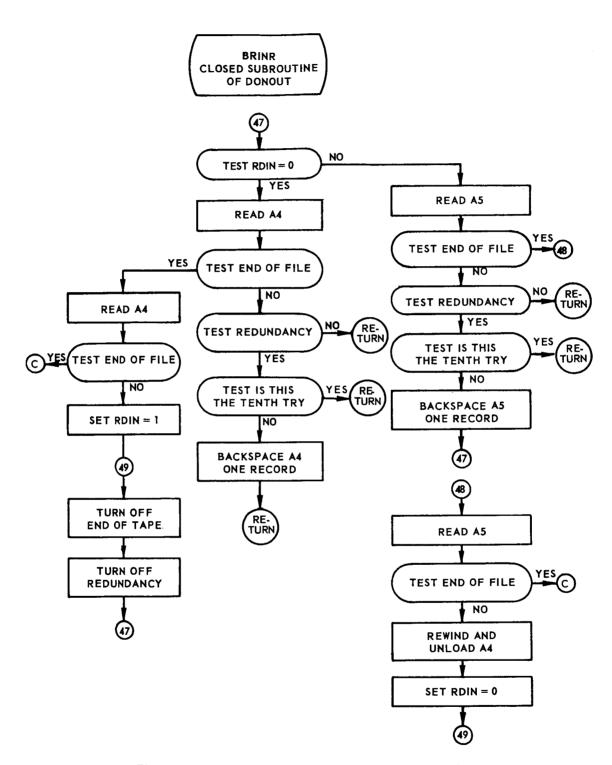


FIGURE 5-15. DONOUT PROGRAM FLOW CHART (Sheet 7 of 7)

5.17 HIGH-SPEED INPUT PROCESSOR PROGRAM (DONIN)

This program processes the high-speed input tape (B4) created by SORTER with data from the Mercury Programming System log tape(s). The position and velocity vectors with an associated vector time and the processed data are placed into COMMON storage for later use in any mission situation that is characterized by associated high-speed input.

The flow chart for DONIN is shown in Figure 5-16.

5.17.1 Input Requirements

Two parameters in the calling sequence of DONIN specify the vector time and data source of a record on the high-speed input tape. In addition to these input parameters (see subsection 4.21.4), the B4 tape written by SORTER is input to DONIN. This program assumes 20-word records on this tape with the following format:

Word 1:	Data Source (an octal integer), i.e.,		
	000 001 000 000 000 002 000 000 000 003 000 000	B-GE Guidance computer IP 7094 computer Raw radar	
Words 2-5:	Remaining ID for message		
Words 6-11:	Telemetry data and time tag		
Word 12:	Vector time (floating-point seconds)		
Words 13-15:	Position vector (floating-point feet)		
Words 16-18:	Velocity vector (floating-point feet/second)		
Words 19-20:	Discrete events data and time tag		

5.17.2 Output Requirements

DONIN reads out the three components of the position vectors and the three components of the velocity vectors with the associated vector time. These are read into COMMON storage locations (see subsection 4.21.4) indicated by the data source.

5.17.3 Method

The high-speed input tape is searched until the data source requested in the call statement is matched by a data source in one of the B4 records. Then, if the time of the high-speed input message being inspected is less than the time requested, control is transferred back to read in the next record. If the time of the message is greater than the time requested, the program backspaces until a message of the correct time and phase is determined. If the beginning of the B4 tape has been reached and the time requested is not found, an error return exit is made from the program.

5.17.4 Usage

Call Statement: a)

> CALL DONIN (TEMPO1, NDATA, NOYES) contains the requested vector time TEMPO1 **NDATA**

is the data source indicator:

1 specifies IP 7094 data 2 specifies B-GE data

3 specifies raw radar data

NOYES

is a return indicator

- 1 indicates an unsuccessful return; a time and data source corresponding to the requested time and data were not found on B4. Vectors and vector time were not transferred to COMMON storage.
- 2 indicates a successful return; a time and data source corresponding to the requested time and data source were found on B4. Vectors and vector time were transferred to COMMON storage.
- Storage—the COMMON storage locations used for the position and b) velocity vectors and vector time are as follows:
 - IP 7094 position and velocity vectors: XIP, YIP, ZIP, XIP1, YIP1, 1) ZIP1.
 - B-GE position and velocity vectors: XI, ETA, ZETA, XII, ETA1, 2) ZETA1.
 - Vector time: T 3)

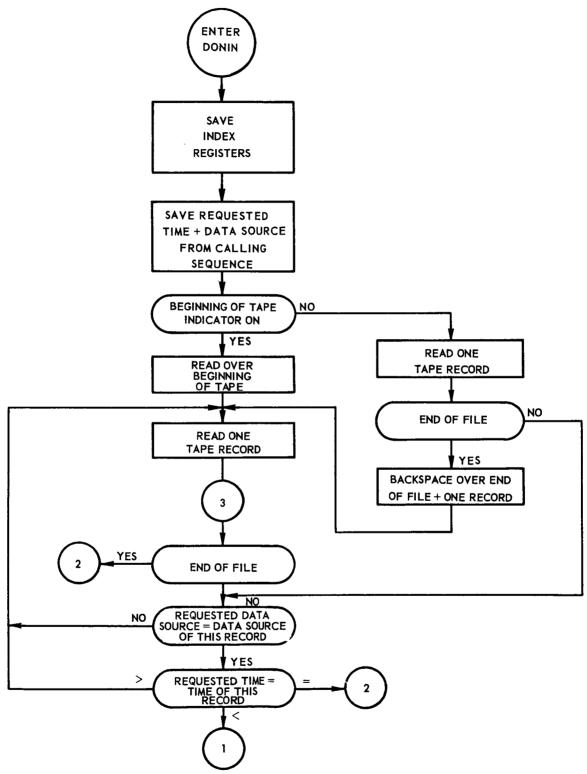


FIGURE 5-16. DONIN PROGRAM FLOW CHART (Sheet 1 of 2)

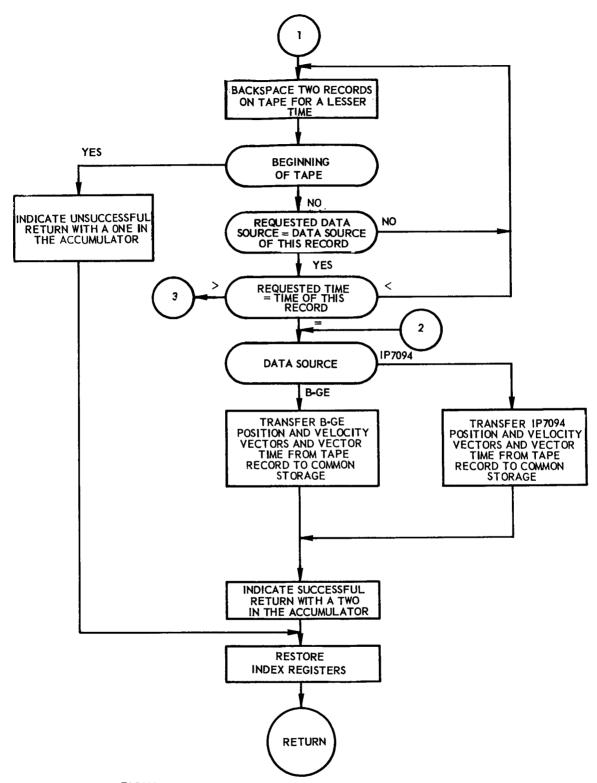


FIGURE 5-16. DONIN PROGRAM FLOW CHART (Sheet 2 of 2)

5.18 LAUNCH PHASE PROCESSOR PROGRAM (LAUNCH)

This program produces launch, abort, and reentry phase parameters for the Postflight Report. This program is used when high-speed input data is available. For the Mercury Atlas (MA) launch phase, this program uses high-speed processed data from either the Impact Predictor 7094 computer or the B-GE computer. LAUNCH also uses high-speed input data for Mercury Redstone (MR) launch and abort situations.

The flow chart for LAUNCH is shown in Figure 5-17.

5.18.1 Input Requirements

Two of the input parameters for this routine—J23, the phase indicator, and NDATA, the input data source indicator—are transferred in the call statement. The rest of the input parameters, along with the physical constants, are placed into COMMON storage prior to entry. This data is used in the equations that produce the output (equations are shown in subsection 4.22.3). In addition, other processor programs are used by LAUNCH. These subordinate programs, which are required for LAUNCH processing, are:

Atmospheric Density Processor Program (ATMOS) B-GE Reference Frame Conversion Program (GECNV) Stagnation Heat Rate Processor Program (HEAT) IP 7094 Reference Frame Conversion Program (IPCNV) True Inertial Coordinate Conversion Program (MERCNV) Range From Launch Pad Processor Program (RFLP)

5.18.2 Output Requirements

The LAUNCH program reads data into COMMON storage before control is returned to Monitor. The computation of these parameters involves the equations shown in subsection 4.22.3. The LAUNCH output parameters are used to write the launch and abort phase paragraphs of the Postflight Report. They are listed below in order of their appearance on the flow chart.

Radial distance of the spacecraft (r) Height above spherical earth ($r - \overline{R}$) Speed in the inertial frame (V_i) Geocentric latitude (L_C) Geodetic latitude (L_D) Height above oblate earth (he) Flight path angle in inertial frame (γ_i) Longitude of spacecraft (λ) Heading angle in inertial frame (ψ_i)

Heading angle in rotational frame (ψ_e) Speed in rotational frame (V_e) Flight path in rotational frame (γ_e) Mach number (M) Reynold's number (R_N) Coefficient of drag (q_D) Local radial distance (r_T)

5.18.3 Method

The equations used to calculate the output of LAUNCH are given in order of their appearance on the flow chart:

$$r = \sqrt{\underline{x}^{2} + \underline{y}^{2} + \underline{z}^{2}}$$

$$r - \overline{R} = (r) - \overline{R}$$

$$V_{i} = \sqrt{\underline{\dot{x}}^{2} + \underline{\dot{y}}^{2} + \underline{\dot{z}}^{2}}$$

$$L_{C} = \tan^{-1} \left[\sqrt{\frac{\underline{Z}}{\underline{x}^{2} + \underline{y}^{2}}} \right]$$

$$L_{D} = \tan^{-1} \left[\left(\frac{a_{c}}{b_{c}} \right)^{2} \tan L_{C} \right]$$

$$e_{2} = 1/(1-f)^{2}$$

$$x_{\nu} = \sqrt{\cos^{2} L_{C} + e_{2} \sin^{2} L_{C}}$$

$$h_{e} = r - x_{\nu}$$

$$\Delta \lambda_{i} = \tan^{-1} (\underline{\overline{y}} / \underline{\overline{x}})$$

$$\gamma_{i} = \sin^{-1} (\dot{x} / V_{i})$$

$$\lambda = \Delta \lambda_i - \omega_e t - \nu_a$$

$$\psi_i = \tan^{-1} (\dot{y}_i / \dot{z})$$

$$\psi_e = \tan^{-1} (\dot{y}/\dot{z})$$
, where $\dot{y} = \dot{y}_i - \omega_e r \cos L_C$

$$V_{e} = \sqrt{\dot{x}^2 + \dot{y}^2 + \dot{z}^2}$$

$$\gamma_{\rm e} = \sin^{-1} (\dot{x}/V_{\rm e})$$

$$M = V_e/C_S$$

$$R_N = V_e / \nu$$

$$q_D = 1/2 \rho v_e^2$$

$$r_{L} = \sqrt{\frac{a_{c}}{\cos^{2} L_{C} + (a_{c}/b_{c})^{2} \sin^{2} L_{C}}}$$

5.18.4 Usage: Call Statement

CALL LAUNCH (J23, DNATA)

The parameter J23 is a phase indicator.

- 1 indicates launch phase (MA and MR)
- 2 indicates abort phase (MR)
- 5 indicates high or medium abort phase (MA)
- 6 indicates low abort phase (MA)

NDATA is a data source indicator.

- 1 indicates IP 7094 data
- 2 indicates B-GE data

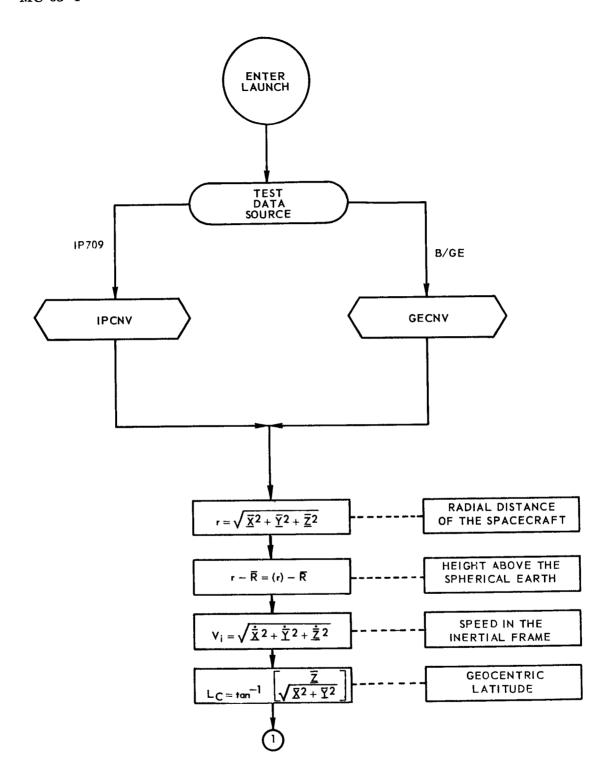


FIGURE 5-17. LAUNCH PROGRAM FLOW CHART (Sheet 1 of 4)

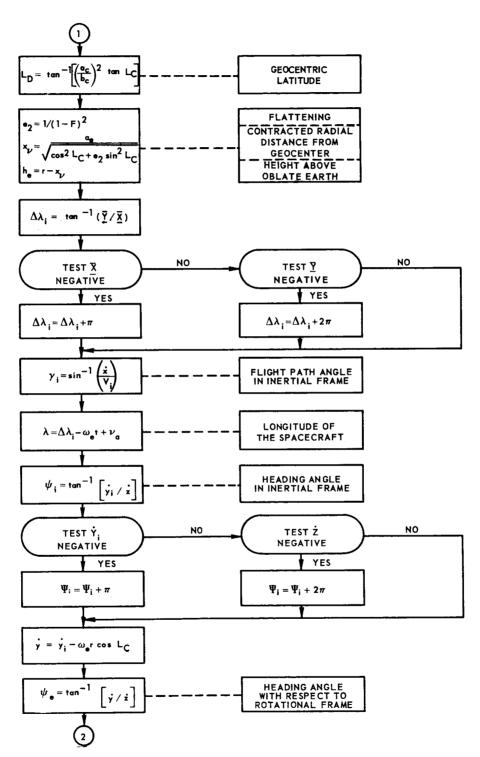


FIGURE 5-17. LAUNCH PROGRAM FLOW CHART (Sheet 2 of 4)

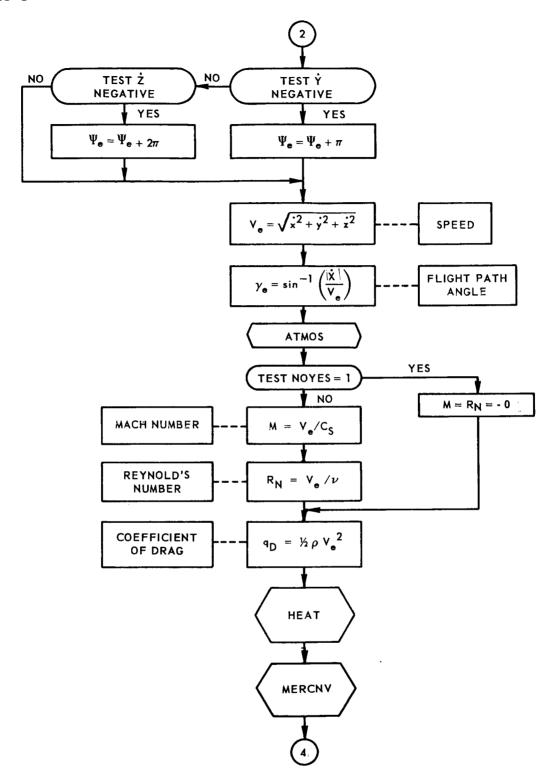


FIGURE 5-17. LAUNCH PROGRAM FLOW CHART (Sheet 3 of 4)

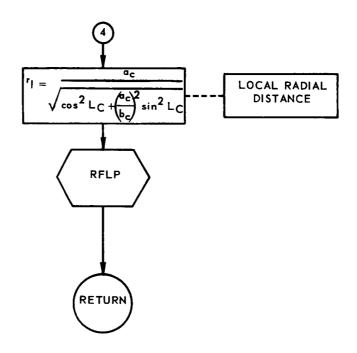


FIGURE 5-17. LAUNCH PROGRAM FLOW CHART (Sheet 4 of 4)

5.19 ORBIT PHASE PROCESSOR PROGRAM (ORBIT)

ORBIT produces orbit phase parameters for the Postflight Report. The program is used where orbit displays are driven (there is no high-speed input).

The flow chart for the ORBIT program is shown in Figure 5-18.

5.19.1 Input Requirements

All input parameters and constants used in the ORBIT output equations are placed into COMMON storage before control is transferred from Monitor.

5.19.2 Output Requirements

ORBIT produces certain orbit phase parameters and places them into COMMON storage prior to its exit. These ORBIT parameters are used to write the orbit phase of the Postflight Report. They are listed below in the order of their appearance on the flow chart.

Radial distance to the spacecraft (r)

Inertial velocity (V_i)

Semi-major axis (a)

Mean motion $(\eta_{\overline{M}})$

Period (T)

Eccentricity (e)

Eccentric anomaly (E)

Semilatus rectum (p)

Inclination angle (i)

Longitude of the ascending node (Ω)

True anomaly (θ_1)

Mean anomaly (Ma)

Elapsed time from perigee passage (t_p)

Argument of perigee (ω)

Height in equatorial plane between X and the projection of the space-craft into that plane $(\Delta \lambda_i)$

Geocentric latitude (L_C)

Height above the oblate earth (he)

Earth-fixed velocity (V2)

Earth-fixed flight-path angle (γ_{e})

Inertial flight-path angle (γ_i)

Earth-fixed longitude (λ)

Inertial heading angle (ψ_i)

Earth-fixed heading angle (ψ_{ρ})

Geodetic latitude (LD)

Height above spherical earth $(r - \overline{R})$

Angular rotation of Ω ($\dot{\Omega}$)

Angular rotation of ω ($\dot{\omega}$)

Semimajor axis less spherical radius (a - \overline{R})

5.19.3 Method

In order of their appearance on the flow chart, the equations required to calculate orbital output parameters are:

$$\mathbf{r} = \sqrt{\underline{\mathbf{x}}^2 + \underline{\mathbf{y}}^2 + \underline{\mathbf{z}}^2}$$

$$\mathbf{v}_{i} = \sqrt{\underline{\dot{\mathbf{x}}}^2 + \underline{\dot{\mathbf{y}}}^2 + \underline{\dot{\mathbf{z}}}^2}$$

$$a = \frac{r}{1 - e \cos(E)}$$

where e cos (E) = $r V_{i}^{2} - 1$

$$\eta_{\text{M}} = \frac{1}{a^{3/2}}$$

$$T = a^{3/2} 2\pi$$

$$e = \sqrt{(e \sin (E))^2 + (e \cos (E))^2}$$

where e sin (E) =
$$\frac{\overline{X} \dot{\overline{X}} + \overline{Y} \dot{\overline{Y}} + \overline{Z} \dot{\overline{Z}}}{\sqrt{a}}$$

E = $\tan^{-1} \left[\frac{e \sin (E)}{e \cos (E)} \right]$

p = $(SP)^2$

where $SP = \sqrt{(\overline{Y} \dot{\overline{Z}} - \dot{\overline{Y}} \underline{Z})^2 + (-\overline{X}\dot{\overline{Z}} + \dot{\overline{X}}\underline{Z})^2 + (\overline{X}\dot{\overline{Y}} - \dot{\overline{X}}\underline{\overline{Y}})^2}$

i = $\tan^{-1} \left[\frac{\sin (i)}{\cos (i)} \right]$

where $RX = \frac{\overline{Y}\dot{\overline{Z}} - \dot{\overline{Y}}\underline{Z}}{SP}$
 $RY = -\frac{\overline{X}\dot{\overline{Z}} + \dot{\overline{X}}\underline{\overline{Z}}}{SP}$
 $RZ = \frac{\overline{X}\dot{\overline{Y}} - \dot{\overline{X}}\underline{Y}}{SP}$

sin (i) = $\sqrt{RX^2 + RY^2}$

cos (i) = RZ

$$\Omega = \tan^{-1} \left[\frac{\sin (\Omega)}{\cos (\Omega)} \right]$$

where sin $(\Omega) = \frac{RX}{\sin (i)}$

cos $(\Omega) = \frac{RY}{\sin (i)}$
 $\theta_1 = \tan^{-1} \left[\frac{\sin (\theta_1)}{\cos (\theta_1)} \right]$

where
$$\cos (\theta_1) = \frac{\frac{P}{r} - 1}{e}$$

$$\sin (E) = \frac{e \sin (E)}{e}$$

$$\cos (E) = \frac{e \cos (E)}{e}$$

$$A = \sin (E)$$

$$B = \cos (E) - e$$

$$QX = (AX BX 3B 3P)$$

$$QY = (AY + BY 3BP)$$

$$QZ = (AZ + BZ) 3BP$$

$$QZ = (AZ + BZ) 3BP$$

$$\sin (\theta_1) = \frac{X QX + Y QY + Z QZ}{r}$$

$$M_a = E - e \sin (E)$$

$$t_p = \frac{T M_a}{2\pi}$$

$$\omega = \tan^{-1} \left[\frac{\sin (\omega)}{\cos (\omega)}\right]$$
where $\sin (\omega) = -QX \cos (\Omega) - QY \sin (\Omega)$

$$A = \frac{\cos (E)}{r}$$

$$B = \sin (E) \sqrt{a}$$

$$PX = AX - BX$$

$$PY = AY - BY$$

$$\begin{array}{rcl} \mathrm{PZ} &=& \mathrm{A} \; \overline{Z} - \mathrm{B} \; \underline{\overline{Z}} \\ \cos \left(\omega \right) = & \mathrm{PX} \cos \left(\Omega \right) + \mathrm{PY} \sin \left(\Omega \right) \end{array}$$
 where $C_1 = 20925672.5$
$$\begin{array}{rcl} \Delta \lambda_i &=& \tan^{-1} \left[\frac{\overline{Z}}{\underline{X}} \right] \\ L_C &=& \tan^{-1} \left[\frac{\overline{Z}}{\sqrt{\underline{X}^2 + \underline{Y}^2}} \right] \end{array}$$

$$\begin{array}{rcl} h_e &=& \mathrm{C}_1 \; \mathrm{r} - \left[\frac{a_e}{\sqrt{\cos^2 \left(L_C \right) + \left(\frac{1}{1 - f} \right)^2 \; \sin^2 \left(L_C \right)}} \right] \end{array}$$

$$\begin{array}{rcl} V_e &=& \sqrt{V \; X^2 + V \; Y^2 + \underline{\dot{Z}}^2} \end{array}$$
 where $VX &=& \underline{\dot{X}} + \underline{\dot{Y}} \; \omega_e \\ VY &=& \underline{\dot{X}} - \omega_e \end{array}$
$$\begin{array}{rcl} VY &=& \frac{\dot{X}}{2} + \underline{\dot{Y}} \; \omega_e \\ VY &=& \frac{\dot{X}}{2} - \omega_e \end{array}$$

$$\begin{array}{rcl} V_e &=& \sin^{-1} \left[\frac{\underline{\dot{X}} \; \underline{\dot{X}} + \underline{\dot{Y}} \; \underline{\dot{Y}} + \underline{\dot{Z}} \; \underline{\dot{Z}}}{\underline{\dot{Y}}} \right] \\ \lambda &=& \mathrm{Sin}^{-1} \left[\frac{\underline{\dot{X}} \; \underline{\dot{X}} + \underline{\dot{Y}} \; \underline{\dot{Y}} + \underline{\dot{Z}} \; \underline{\dot{Z}}}{\underline{\dot{Y}}} \right] \\ \lambda &=& \Delta \lambda_i - \omega_t - \upsilon_a \end{array}$$

$$\psi_i &=& \cos^{-1} \left[\frac{\mathrm{r} \; \underline{\dot{Z}} - \underline{\ddot{Z}} \; V_i \; \sin \left(\gamma_i \right)}{V_i \; \cos \left(\gamma_i \right) \; \sqrt{\dot{X}^2 + \underline{\dot{Y}}^2}} \right]$$

$$\psi_{e} = \cos^{-1} \left[\frac{r \, \dot{\overline{Z}} - \overline{Z} \, V_{e} \, \sin \left(\gamma_{e} \right)}{V_{e} \, \cos \left(\gamma_{e} \right) \, \sqrt{\overline{X}^{2} + \overline{Y}^{2}}} \right]$$

$$L_{D} = \tan^{-1} \left[\frac{TAN \, (L_{C})}{\left(\frac{b_{c}}{a_{c}} \right)^{2}} \right]$$

$$(r - \overline{R}) = r - \overline{R}$$

$$\dot{\Omega} = \frac{-C_{2} \, \eta_{M} \, C_{3} \, \cos \left(i \right) \, T}{C_{A}}$$

where
$$C_2 = 1623.48 \times 10^{-6}$$

$$C_3 = \left(\frac{C_1}{a}\right)^2$$

$$c_4 = (1 - e)^2$$

$$\dot{\omega} = \frac{c_2}{2} \eta_{M} \frac{c_3}{c_4} (5 \cos^2 (i) - 1) T$$

$$(a - \overline{R}) = a - \overline{R}$$

5.19.4 Usage: Call Statement

CALL ORBIT

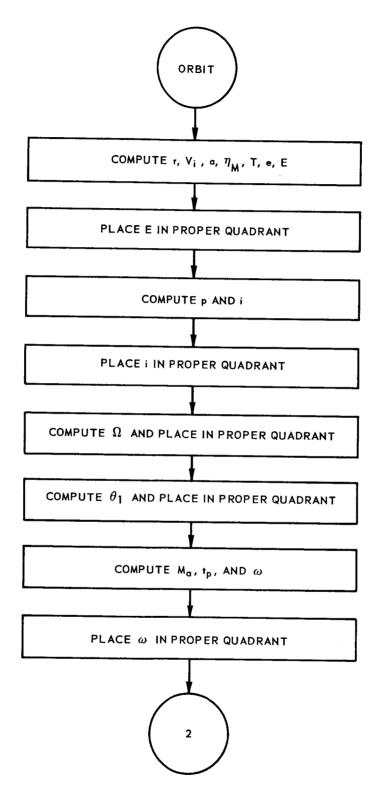


FIGURE 5-18. ORBIT PROGRAM FLOW CHART (Sheet 1 of 2)

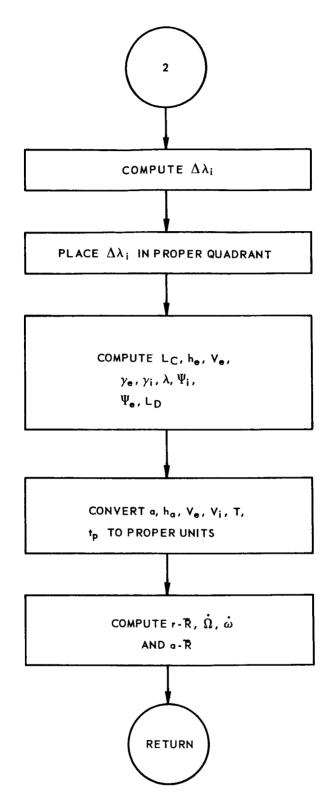


FIGURE 5-18. ORBIT PROGRAM FLOW CHART (Sheet 2 of 2)

5.20 REENTRY PHASE PROCESSOR PROGRAM (RENTER)

RENTER produces reentry phase parameters for the Postflight Report. This program is used when reentry displays are driven (there is no associated high-speed input). It is also used when abort displays are driven and there is no associated high-speed input.

The flow chart for the RENTER program is shown in Figure 5-19.

5.20.1 Input Requirements

The phase indicator J23 is transferred in the call statement. All other input parameters and constants used in the RENTER equations are placed into COMMON storage prior to entry of the program. These equations produce output for RENTER. This program also uses other processor programs, which are:

Atmospheric Density Processor Program (ATMOS)
Stagnation Heat Rate Processor Program (HEAT)
True Inertial Coordinate Conversion Program (MERCNV)
Range From Launch Pad Processor Program (RFLP)

5.20.2 Output Requirements

RENTER outputs data for a phase indicator in its call statement (see subsection 4.24.4). The program also reads out reentry parameters that are placed into COMMON storage before its exit. The computation of these parameters involves the equations shown in subsection 4.24.3. The RENTER output parameters are used to write the reentry phase and sometimes the abort phase paragraph of the Postflight Report. They are listed below in order of their appearance on the flow chart.

Geocentric radial distance to the spacecraft (r)

Inertial velocity (Vi)

Angle in equatorial plane between X and the projection of the space-craft into that plane $(\Delta \lambda_i)$

Geocentric latitude (L_C)

Height above the oblate earth (he)

Earth-fixed velocity (Ve)

Earth-fixed flight-path angle $(\underline{\gamma}_e)$

Inertial flight-path angle (γ_i)

Inertial heading (ψ_i)

Earth-fixed heading (ψ_{α})

Earth-fixed longitude (λ)

Geodetic latitude (LD)

Height above the spherical earth $(r - \overline{R})$

Mach number (M)

Reynolds number (R_N)

Coefficient of drag (QD)

Local radius of the earth (r₁)

Reentry range from retrofire (S_R)

5.20.3 Method

The equations used to calculate the output parameters for the RENTER are shown below, in order of their appearance on the flow chart.

$$r = \sqrt{\underline{X}^2 + \underline{Y}^2 + \underline{Z}^2}$$

$$V_i = \sqrt{\underline{\dot{X}}^2 + \underline{\dot{Y}}^2 + \underline{\dot{Z}}^2}$$

$$\Delta \lambda_i = \tan^{-1} \left(\frac{\underline{\underline{Y}}}{\underline{\underline{X}}} \right)$$

$$L_C = \tan^{-1} \left[\frac{\underline{Z}}{\underline{\underline{X}}^2 + \underline{\underline{Y}}^2} \right]$$

$$h_e = r C_1 - \left[\frac{a_e}{\sqrt{\cos^2 (L_C) + \left(\frac{1}{1 - f}\right)^2 \sin^2 (L_C)}} \right]$$
where $C_1 = 20925672.5$

$$V_e = \sqrt{V X^2 + V Y^2 + \underline{\dot{Z}}^2}$$

where
$$VX = \frac{\dot{X}}{X} + \frac{Y}{Y} \omega_{e}$$

$$VY = \dot{Y} - X \omega_{e}$$

$$\gamma_{e} = \sin^{-1} \left[\frac{\frac{X}{X} VX + \frac{Y}{Y} VY + \frac{Z}{Z} \frac{\dot{Z}}{Z}}{V_{e}} \right]$$

$$\gamma_{i} \qquad \sin^{-1} \left[\frac{\frac{X}{X} + \frac{Y}{Y} + \frac{Z}{Z} \frac{\dot{Z}}{Z}}{V_{i}} \right]$$

$$\psi_{e} = \cos^{-1} \left[\frac{r \frac{\dot{Z}}{Z} - \frac{Z}{Z} V_{i} \sin{(\gamma_{i})}}{V_{i} \cos{(\gamma_{i})} \sqrt{X^{2} + \frac{Y}{Y^{2}}}} \right]$$

$$\lambda = \cos^{-1} \left[\frac{r \frac{\dot{Z}}{Z} - \frac{Z}{Z} V_{e} \sin{(\gamma_{e})}}{V_{e} \cos{(\gamma_{e})} \sqrt{X^{2} + \frac{Y}{Y^{2}}}} \right]$$

$$\lambda = \Delta \lambda_{i} - \omega_{e} t - \nu_{a}$$

$$L_{D} = \tan^{-1} \left[\frac{\tan{(L_{C})}}{\left(\frac{b_{C}}{a_{C}}\right)^{2}} \right]$$

$$(r - \overline{R}) = r - \overline{R}$$

$$M = \frac{V_{e}}{C_{S}}$$

 $R_N = \frac{V_e}{V}$

$$q_{D} = 1/2 \rho V_{e}^{2}$$

$$r_{e} = \frac{a_{e}}{\sqrt{\cos^{2}(L_{C}) + \frac{\sin^{2}(L_{C})}{\left(\frac{b_{c}}{a_{c}}\right)^{2}}}}$$

$$S_{R} = 1/2 (r_{1} + r_{1 B}) \cos^{-1} \left[\cos(L_{C} - L_{C B})\right]$$

$$- 2 \sin^{2} \left(\frac{\lambda - \lambda_{B}}{2}\right) \cos(L_{C}) \cos(L_{C B})$$

5.20.4 Usage: Call Statement

CALL RENTER (J 23)

J 23 is a phase indicator

- 4 indicates reentry phase
- 5 indicates high abort phase (MA)
- 6 indicates low abort phase (MA)

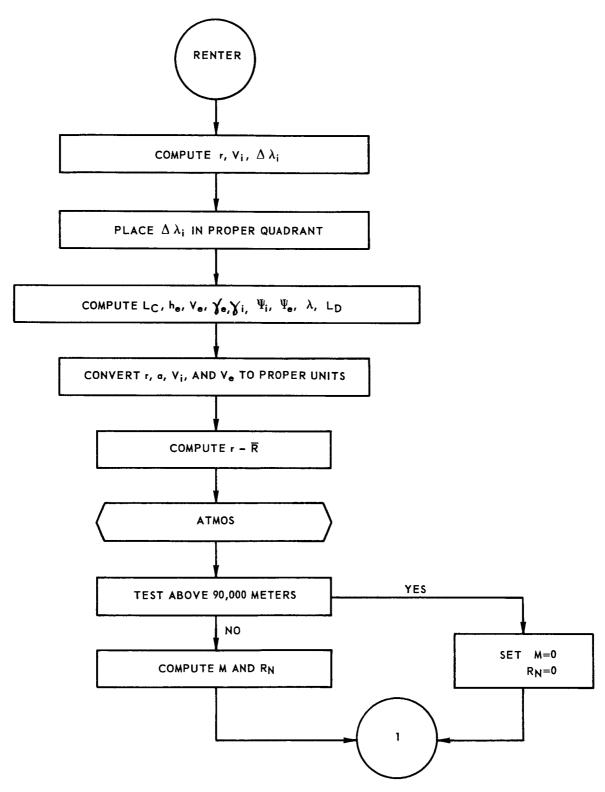


FIGURE 5-19. RENTER PROGRAM FLOW CHART (Sheet 1 of 2)

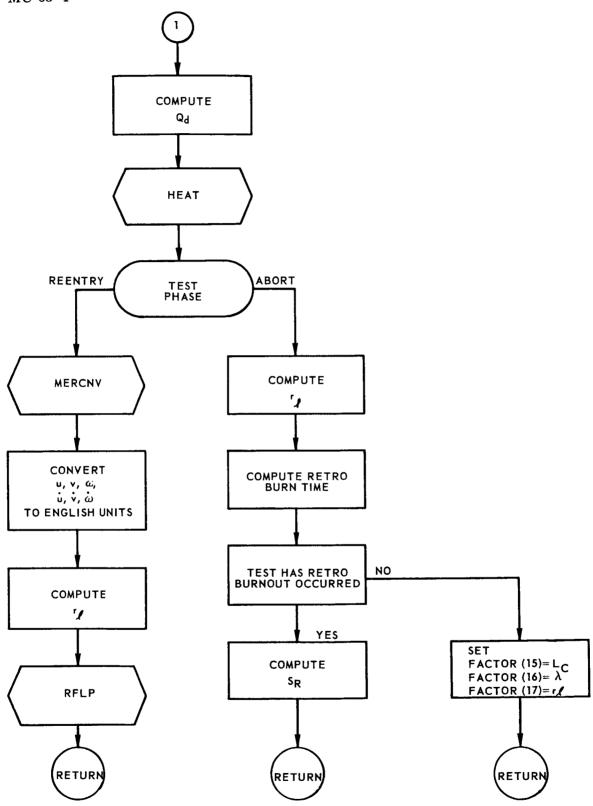


FIGURE 5-19. RENTER PROGRAM FLOW CHART (Sheet 2 of 2)

5.21 NUMERICAL INTEGRATION PROGRAM (NUMIN)

The numerical integration program is used during the orbit and reentry phases to compute position and velocity vectors along the trajectory. The integration within this program is done by the NOCPNI subroutine used by the Goddard operational Mercury programs.

A flow chart of NUMIN is given in Figure 5-20.

5.21.1 Input Requirements

The input parameters used by NUMIN are transferred through the CALL statement.

CALL NUMIN (MPHASE, IERROR, TT, XX, YY, ZZ, XXD, YYD, ZZD, TTL)

 $MPHASE = \frac{1 \text{ for orbit phase}}{2 \text{ for reentry phase}}$

TT = anchor time of input vector

XX = X coordinate of input position vector

YY = Y coordinate of input position vector

ZZ = Z coordinate of input position vector

 $XXD = \dot{X}$ component of input velocity vector

 $YYD = \dot{Y}$ component of input velocity vector

 $ZZD = \dot{Z}$ component of input velocity vector

TTL = final time of the period over which the integration is to take place.

5.21.2 Output Requirements

The output from NUMIN is a table of position and velocity vectors, and an indication as to whether the integration was successful. The vector table is set up as follows:

TNINT = time increment of table (0 = minutes; 1 = seconds)

+1 = integration output interval

+2 = time of first vector in table

+3 = time of last vector in table

+4
+5
+6
+7
+8
+9
= coordinates of first position vector

rec., each subsequent position and velocity vector will now follow.

The integration indicator is transferred through the call statement and is set as follows:

IERROR = 1 for successful integration

2 for unsuccessful integration

5.21.3 Method

The numerical integration method used in this routine is described in the NOCPNI program write-up. The NUMIN routine merely sets up the input data and calls NOCPNI.

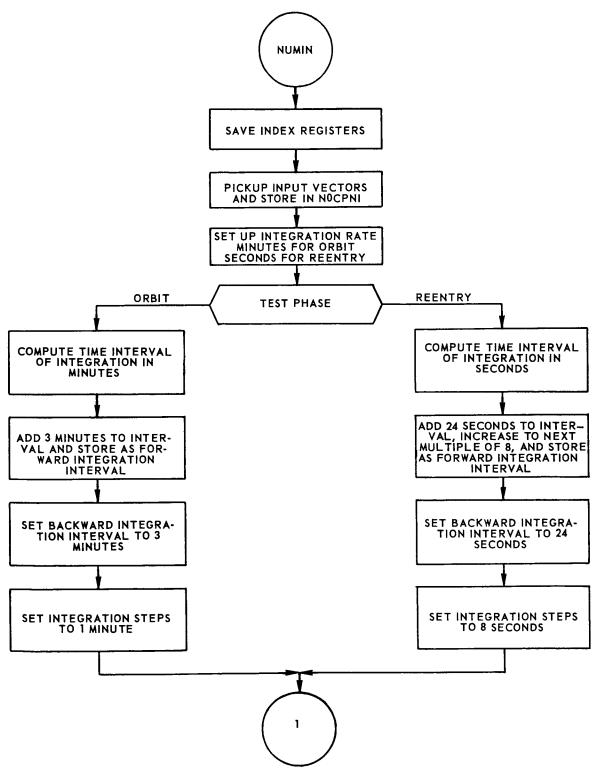


FIGURE 5-20. NUMIN PROGRAM FLOW CHART (Sheet 1 of 2)

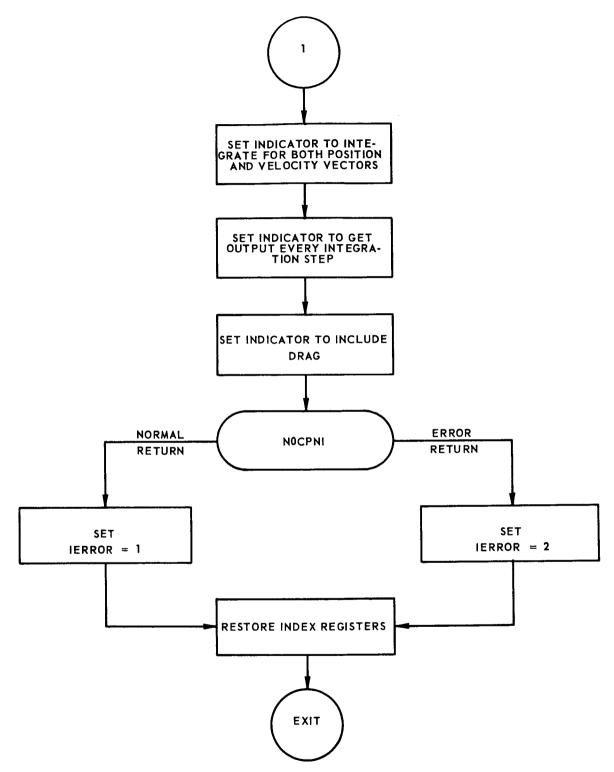


FIGURE 5-20. NUMIN PROGRAM FLOW CHART (Sheet 2 of 2)

5.22 IP 7094 REFERENCE FRAME CONVERSION PROGRAM (IPCNV)

This subroutine to the LAUNCH program converts IP 7094 position (\vec{r}) and velocity (\vec{v}) components (X, Y, Z), or processed data, to true inertial coordinates $(\underline{X}, \overline{Y}, \overline{Z})$. Input to IPCNV is placed into COMMON storage before entry. The program is entered with the call statement CALL IPCNV.

The flow chart for IPCNV is shown in Figure 5-21.

The conversion is performed in the following manner. If M is the transformation matrix

$$\mathbf{M} = \begin{pmatrix} \cos(\nu_{\mathbf{a}} + \omega_{\mathbf{e}} \mathbf{t}) & -\sin(\nu_{\mathbf{a}} + \omega_{\mathbf{e}} \mathbf{t}) & 0 \\ \sin(\nu_{\mathbf{a}} + \omega_{\mathbf{e}} \mathbf{t}) & \cos(\nu_{\mathbf{a}} + \omega_{\mathbf{e}} \mathbf{t}) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

then

$$\vec{r} = \begin{pmatrix} \frac{\overline{X}}{\underline{Y}} \\ \frac{\overline{Y}}{\underline{Z}} \end{pmatrix} = M \cdot \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \qquad \vec{v} = \frac{\dot{\underline{Y}}}{\underline{Y}} = M \cdot \begin{pmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{pmatrix}$$

Prior to its exit, IPCNV places its output into COMMON storage.

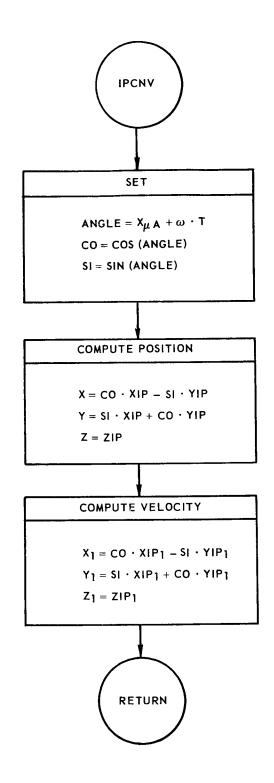


FIGURE 5-21. IPCNV PROGRAM FLOW CHART

5.23 B-GE REFERENCE FRAME CONVERSION PROGRAM (GECNV)

GECNV is a subroutine of the LAUNCH program. It converts processed data, position (\vec{r}) and velocity (\vec{v}) components from the B-GE reference frame (ξ, η, ζ) to true inertial coordinates $(\underline{X}, \underline{Y}, \underline{Z})$. Input to GECNV is placed in COMMON storage before entry. The program is entered with the call statement CALL GECNV.

The flow chart for GECNV is shown in Figure 5-22.

The conversion is performed in the following manner. If N is the transformation matrix

$$N = \begin{pmatrix} \cos \left(\delta \varphi_{R} + \omega_{e} t\right) & -\sin \left(\delta \varphi_{R} + \omega_{e} t\right) & 0 \\ \sin \left(\delta \varphi_{R} + \omega_{e} t\right) & \cos \left(\delta \varphi_{R} + \omega_{e} t\right) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

then

$$\vec{r} = \begin{pmatrix} \overline{\underline{X}} \\ \overline{\underline{Y}} \\ \overline{\underline{Z}} \end{pmatrix} = N \cdot \begin{pmatrix} \xi \\ \eta \\ \zeta \end{pmatrix}$$

$$\vec{v} = \begin{pmatrix} \frac{\dot{x}}{X} \\ \frac{\dot{y}}{Z} \end{pmatrix} = N \cdot \begin{pmatrix} \dot{\xi} \\ \dot{\eta} \\ \dot{\zeta} \end{pmatrix}$$

Prior to its exit, GECNV places its output in COMMON storage.

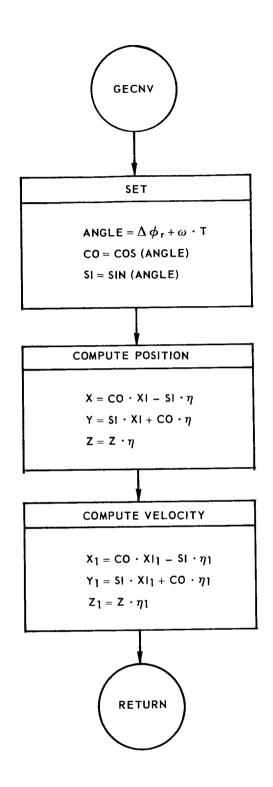


FIGURE 5-22. GECNY PROGRAM FLOW CHART

5.24 TRUE INERTIAL COORDINATE CONVERSION PROGRAM (MERCNV)

MERCNV, a subroutine of LAUNCH, converts position (\vec{r}) and velocity (\vec{v}) components in the true inertial coordinate frame $(X, \overline{Y}, \overline{Z})$ to pad rectangular coordinates (u, v, w). All input quantities are placed in COMMON storage prior to entry with the call statement, CALL MERCNV.

The flow chart for MERCNV is shown in Figure 5-23.

The sequence of transformation for \vec{r} is

$$\begin{pmatrix} \xi_{\rho} \\ \eta_{\rho} \end{pmatrix} = \begin{pmatrix} \cos (\delta \varphi_{p} + \omega_{e} t) & \sin (\delta \varphi_{p} + \omega_{e} t) & 0 \\ -\sin (\delta \varphi_{p} + \omega_{e} t) & \cos (\delta \varphi_{p} + \omega_{e} t) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \overline{\underline{X}} \\ \overline{\underline{Y}} \\ \overline{\underline{Z}} \end{pmatrix}$$

$$\begin{pmatrix} U' \\ V' \end{pmatrix} = \begin{pmatrix} \eta_{\rho} \\ \zeta \end{pmatrix} = \begin{pmatrix} 0 \\ r' \cdot \sin V \\ \overline{\zeta} & \sin V \end{pmatrix}$$

$$\begin{pmatrix} \mathbf{U}' \\ \mathbf{V}' \\ \mathbf{W}' \end{pmatrix} = \begin{pmatrix} \eta_{\rho} \\ \zeta_{\rho} \\ \xi_{\rho} \end{pmatrix} - \begin{pmatrix} 0 \\ \mathbf{r'}_{0} \sin L_{\rho} \\ \mathbf{r'}_{0} \cos L_{\rho} \end{pmatrix}$$

$$\begin{pmatrix} \overline{U} \\ \overline{V} \\ \overline{W} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 \cos L_{p'} - \sin L_{p'} \\ 0 \sin L_{p'} \cos L_{p'} \end{pmatrix} \qquad \begin{pmatrix} U' \\ V' \\ W' \end{pmatrix}$$

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} \sin\Theta_0 & \cos\Theta_0 & 0 \\ -\cos\Theta_0 & \sin\Theta_0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \begin{pmatrix} \overline{U} \\ \overline{V} \\ \overline{W} \end{pmatrix}$$

The sequence for \vec{v} is:

$$\begin{pmatrix} \dot{\xi} \\ \rho \\ \dot{\eta} \\ \rho \\ \dot{\zeta} \\ \rho \end{pmatrix} = \begin{pmatrix} \cos \left(\delta\Theta_{p} + \omega_{e} t\right) & \sin \left(\delta\Theta_{p} + \omega_{e} t\right) & 0 \\ -\sin \left(\delta\Theta_{p} + \omega_{e} t\right) & \cos \left(\delta\Theta_{p} + \omega_{e} t\right) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \dot{\underline{X}} + \omega_{e} \\ \dot{\underline{Y}} \\ \dot{\underline{Y}} - \omega_{e} \\ \dot{\underline{X}} \end{pmatrix}$$

$$\begin{pmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{pmatrix} = \begin{pmatrix} \sin \Theta_{0} & \cos \Theta_{0} & 0 \\ -\cos \Theta_{0} & \sin \Theta_{0} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos L_{p} \\ -\sin L_{p} \\ \cos L_{p} \end{pmatrix} \begin{pmatrix} \dot{\eta}_{\rho} \\ \dot{\zeta}_{\rho} \\ \dot{\xi}_{\rho} \end{pmatrix}$$

MERCNV places its output values into COMMON storage prior to its exit.

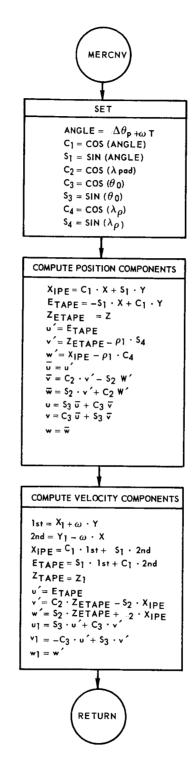


FIGURE 5-23. MERCNY PROGRAM FLOW CHART

5.25 ATMOSPHERIC DENSITY PROCESSOR PROGRAM (ATMOS)

This subroutine of LAUNCH and RENTER produces, in English units, the atmospheric density, kinematic viscosity, and speed of sound at spacecraft position. ATMOS prepares for aerodynamic parameter calculations the mach number (ratio of relative speed to local speed of sound), Reynold's number per foot (ratio of relative speed to local kinematic viscosity), and dynamic pressure $(V_{\rm e}^{\ 2}/_2)$.

The ATMOS flow chart is shown in Figure 5-24.

Input to ATOMS, including the coefficients of density polynomials set up by INITIA, are in COMMON storage before entry. Control is transferred to ATMOS with the call statement CALL ATMOS (KERR). The return indicator, KERR, is part of the output from the program. The remaining output is placed in COMMON before the program exits.

KERR = 1 when the height of the spacecraft exceeds 90 geopotential kilometers, in which case the speed of sound ceases to be uniquely defined independent of sonic frequency. Therefore, kinematic viscosity and the speed of sound (actually, the mach number and Reynold's number) are meaningless and are set to minus zero (-0). Alternatively, KERR = 2 if both kinematic viscosity and the speed of sound are calculated.

The equations used by ATMOS are given below:

$$h = \frac{a_e h}{a_e + h}$$
 meters converted to geopotential meters

The seventh degree polynomials of atmospheric density are:

$$\begin{split} L_{n} (\rho) &= \sum_{i=0}^{7} \quad a_{i} \quad \left(\frac{h}{10^{5}}\right) \quad i \quad h < 136025.0 \\ &= \sum_{i=0}^{7} \quad b_{i} \quad \left(\frac{h}{10^{5}}\right) \quad i \quad h \stackrel{\geq}{=} 136025.0 \\ &\rho = \exp \left[L_{n} (\rho)\right] \quad (\text{kilograms/meters}^{3}) \end{split}$$

$$T_M = (T_M)_b + L_M (h - h_b)$$
 for proper base value, b.

$$C_S = 20.046333 \sqrt{T_M}$$
 (meters/second)

$$\nu = \frac{1.458 \times 10^{-6} \times T_{\text{M}}^{3/2}}{(T_{\text{M}} + 110.4)}$$
 (meters²/second)

converting $\nu\text{, }\mathbf{C}_{S}$ and ρ to English units:

$$\nu$$
 (ft. 2 /sec.) = ν /. 9290304

$$C_{S}(ft./sec.) = C_{S}/.3048$$

$$\rho \text{ (slug/ft.}^3) = \rho/515.378725$$

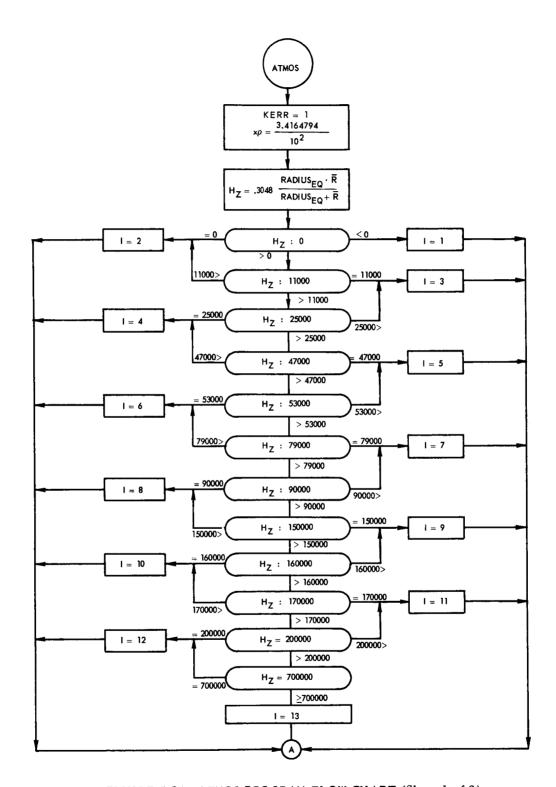


FIGURE 5-24. ATMOS PROGRAM FLOW CHART (Sheet 1 of 2)

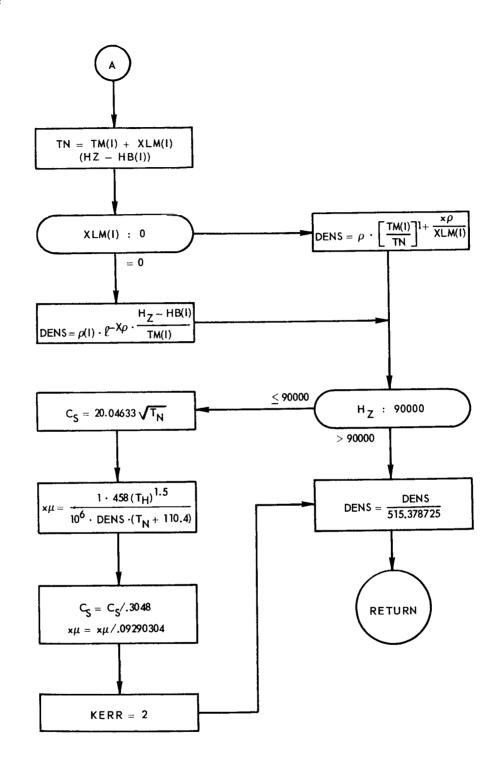


FIGURE 5-24. ATMOS PROGRAM FLOW CHART (Sheet 2 of 2)

5.26 STAGNATION HEAT RATE PROCESSOR PROGRAM (HEAT)

HEAT calculates the stagnation heat rate, in English units, needed by LAUNCH and RENTER. Inputs to HEAT are placed into COMMON storage before entry. Transfer of control to the program is effected by the call statement, CALL HEAT. The program places its single output, $\mathbf{q}_{\mathbf{S}}$ (BTU/sec.ft.), into COMMON before its exit.

The flow chart for HEAT is shown in Figure 5-25.

The methodology involved in calculating this data is shown below:

If $\rho_0 = \text{sea level density (slugs/foot}^3)$

 μ_{∞} = free stream speed (feet/second, relative)

 $h_s = \text{stagnation enthalpy (BTU/pound)}$

 h_{w} = wall enthalpy (BTU/pound f(t)(p))

 h_{w540} = reference wall enthalpy (=130 BTU/pound)

$$\sqrt{R} = 2.58 \text{ (foot}^{1/2}\text{)}$$

Then, in continuum flow (h \leq 350,000 feet)

$$q_S = \epsilon \sigma T^4$$
, where $\epsilon = 0.8$ and $\sigma = \frac{0.171 \times 10^8}{3600}$ BTU/hour

and in free molecular flow (h > 350,000 feet)

$$q_s = 2.69 \times 10^7 \quad \eta \left(\frac{\rho_{\infty}}{\rho_o}\right) \left(\frac{\mu_{\infty}}{\mu_c}\right)^3$$
 (BTU/feet² per second)

where ρ_{∞} = free stream density (slugs/foot³)

 $\mu_{\rm o}$ = spacecraft speed 26,000 (feet/second, inertial)

 $\eta = 1.0$

From these equations, then, the following can be stated:

$$q_S = 4.879 \sqrt{\rho} U_D^{3.15}, h \le 350,000 \text{ feet}$$

$$q_s = 6.439 \times 10^5 \rho U_D^3$$
, $h > 350,000 \text{ feet}$

where UD = free stream speed (kilofeet/second, earth-fixed)

 ρ = free stream density (slug/foot³)

q_s = stagnation heating rate (BTU/seconds per foot²)

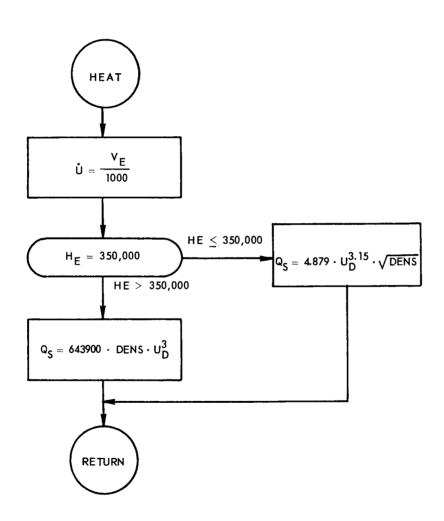


FIGURE 5-25. HEAT PROGRAM FLOW CHART

5.27 RANGE FROM LAUNCH PAD PROCESSOR PROGRAM (RFLP)

RFLP, a LAUNCH program subroutine, calculates S, the range from the launch pad of the spacecraft. Inputs are placed in COMMON storage before the program is called with the statement CALL RFLP. This program uses the utility routine ARCCOS in calculating S which is the only output of RFLP. S is placed into COMMON prior to the exit of RFLP. The equation for range from launch pad is as follows:

$$S = B \cdot \left(\frac{\overline{R}_p + R_L}{2}\right)$$
 where $\frac{\overline{R}_p + R_L}{2}$ is an average radius

and $B = \cos^{-1}(A)$. If |A| > 1, then S is set equal to zero and the program returns control to Monitor. If $|A| \le 1$, then S is calculated as follows:

$$S = \cos^{-1} \left(\sin L_D \sin L'_p \cos L_D \cos L'_p \cos (\lambda - \lambda_1) \right) \cdot \left(\frac{\overline{R}_p + R_L}{2} \right)$$

before control returns to Monitor.

The RFLP flow chart is shown in Figure 5-26.

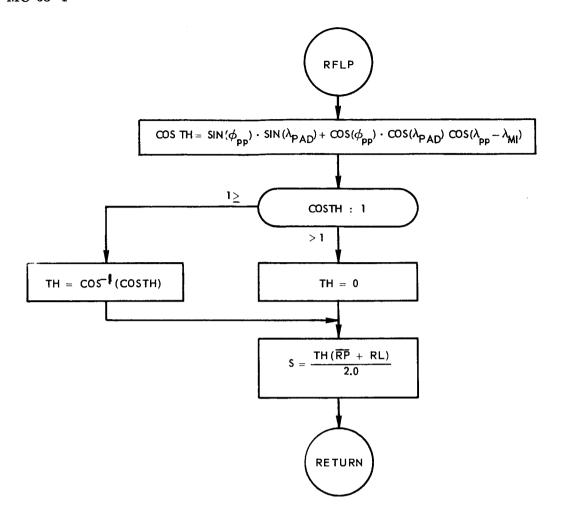


FIGURE 5-26. RFLP PROGRAM FLOW CHART

5.28 LANGRANGIAN INTERPOLATION (GRUNGY)

The Langrangian interpolation program is used during the orbit and reentry phases to interpolate for position and velocity vectors at any time in the trajectory. The interpolation within this program is done by the U7INTP subroutine used by the Goddard operational Mercury programs. A complete writeup of U7INTP is given in subsection 1.3.15 so it will only be referenced here.

A flow chart of GRUNGY is given in Figure 5-27.

5.28.1 Input Requirements

The input parameters used by GRUNGY are transferred through the CALL statement and supplied by an integration table produced by NUMIN.

CALL GRUNGY (TTN, XXN, YYN, ZZN, XXDN, YYDN, ZZDN, IERROR)

TTN = Time of the desired vector in GMT seconds.

5.28.2 Output Requirements

The output from GRUNGY is a position and velocity vector which is transferred through the CALL statement

XXN = X coordinate of the output position vector

YYN = Y coordinate of the output position vector

ZZN = Z coordinate of the output position vector

 $XXDN = \dot{X}$ component of the output velocity vector

 $YYDN = \dot{Y}$ component of the output velocity vector

 $ZZDN = \dot{Z}$ component of the output velocity vector

IERROR = 1 successful interpolation

2 unsuccessful interpolations

5.28.3 Method

The Langrangian interpolation method used in this routine is described in subsection 1.3.15. The GRUNGY routine merely sets up the input data and calls U7INTP.

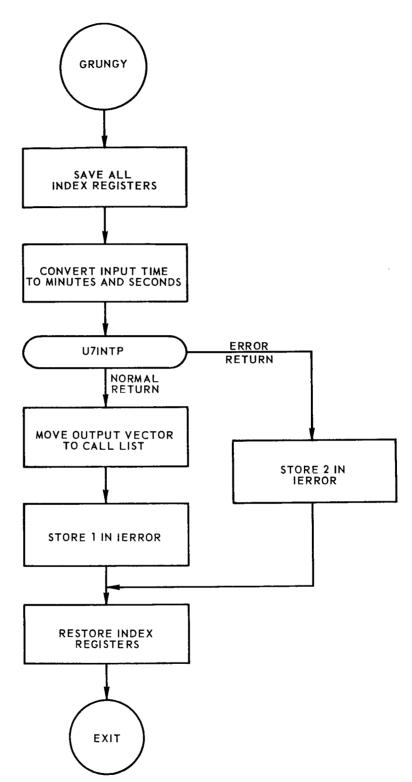


FIGURE 5-27. GRUNGY PROGRAM FLOW CHART

5.29 RECOVERY AREA CONVERSION ROUTINE (RCACNV)

RCACNV converts the recovery area numeric code to a recognizable alphanumeric code.

A flow chart of RCACNV is shown in Figure 5-28.

5.29.1 Input Requirements

The numeric value of the recovery area is in the calling sequence (JAREA).

5.29.2 Output Requirements

The alphanumeric value is replaced in the calling sequence (RECREA).

5.29.3 Method

The program acquires the numeric value of the recovery area from the call sequence (JAREA) and places the number in Index Register 1. The program then uses the Index Register to find the corresponding alphanumeric code. It stores this code back in the calling sequence (RECREA) and returns to Monitor.

5.29.4 Usage: Call Statement

Call RCACNV (JAREA, RECREA).

JAREA is the numeric code for the recovery area. RECREA is the corresponding alphanumeric code for the recovery area. (See key to Quick Look or Three-Day Report for conversions.)

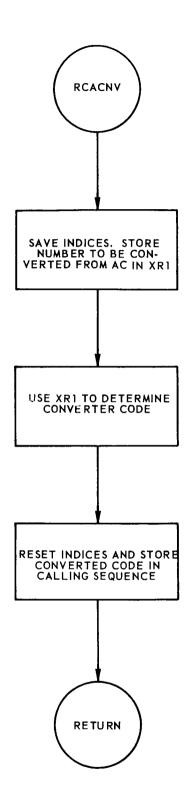


FIGURE 5-28. RCACNV PROGRAM FLOW CHART

5.30 DISTANCE COMPUTATION OF EARTH AND SPACE TRACK (DISTAN)

DISTAN computes the projected earth-surface distance and space-track distance traveled by the spacecraft from liftoff to impact.

The flow chart for DISTAN is shown in Figure 5-29.

5.30.1 Input Requirements

The following parameters are transferred in the call statement:

- a) Present position, φ and λ .
- b) Time associated with present position.
- c) The height of the spacecraft above the oblate earth.
- d) The phase indicator.

5.30.2 Output Requirements

The following data on A3 is the output from DISTAN:

- a) The last anchor time of the arc length.
- b) The arc lengths for the earth and space tracks.
- c) The sums of the arc lengths for the earth and space tracks associated with the anchor time.

5.30.3 Method

The following formulas are used to calculate the distance, D.

a)
$$D_i = \sum_{t=0}^{n} \Delta \ell_i$$
 where $i = 1$ or 2 $n = total time of flight$

b)
$$\Delta \ell_i = R_i \sqrt{(\Delta \varphi)^2 + [(\Delta \lambda) (\cos \varphi_B)]^2}$$

c)
$$R_1 = A \sqrt{\frac{1}{\cos^2 \varphi_B + \frac{\sin^2 \varphi_B}{1 - \ell_1^2}}}$$

d)
$$R_2 = (A + H) \sqrt{\frac{1}{\cos^2 \varphi_B + \frac{\sin^2 \varphi_B}{1 - \ell_2^2}}}$$

where

$$\Delta \varphi = \varphi_{B} - \varphi_{A}$$

$$\Delta \lambda = \lambda_B - \lambda_A$$

D₁ = total distance traveled by the spacecraft projected on earth's surface

 D_2 = total space track distance

 $\Delta \ell$ = arc length from time t - Δt to t

A = semimajor axis of oblate earth

 ℓ_1 = eccentricity of elliptic earth

 ℓ_2 = eccentricity of orbital ellipse

 φ_A , λ_A = present position at time t - Δt

 φ_{B} , λ_{B} = present position at time t

5.30.4 Usage: Call Statement

CALL DISTAN (IIFT, LXED, CLED, TIMNS, HAOE, NFASE)

IIFT = 0 first time through routine

 \neq 0 not first time through

XLED = longitude of present position

CLED = geocentric latitude of present position

TIMNS = time of present position in seconds

HAOE = height above oblate earth

NFASE = phase indicator

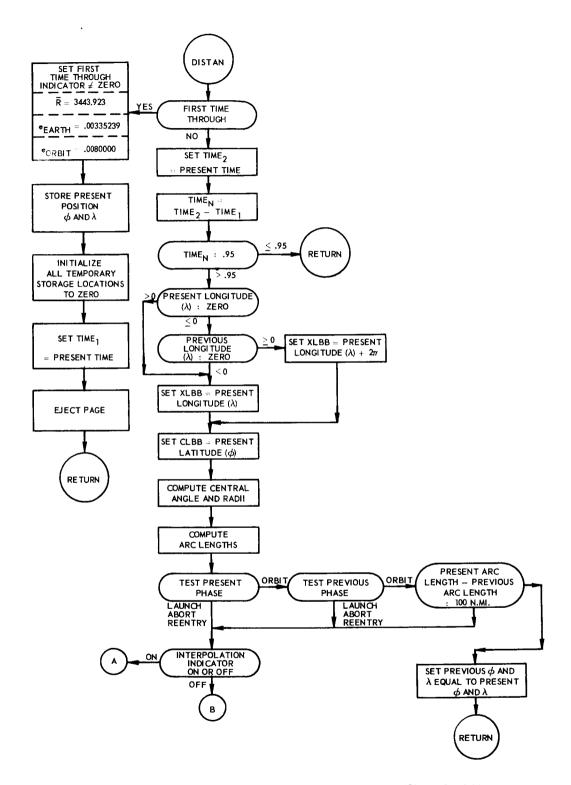


FIGURE 5-29. DISTAN PROGRAM FLOW CHART (Sheet 1 of 2)

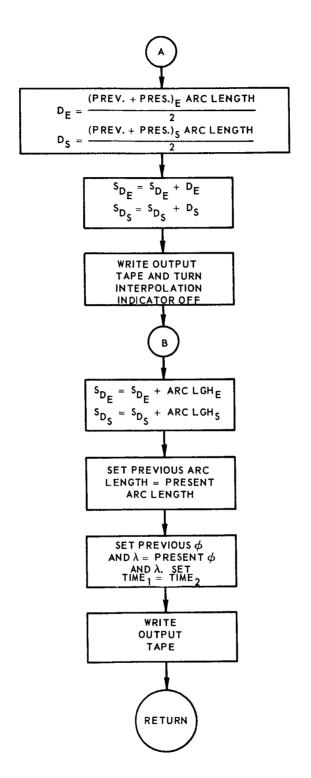


FIGURE 5-29. DISTAN PROGRAM FLOW CHART (Sheet 2 of 2)

5.31 UTILITY PROGRAMS

The Postflight Reporter program has eight subroutines which are of a utility nature. These programs are used so frequently in Monitor and other Postflight Reporter programs that they merit separate definition.

The flow charts for the utility programs are shown in Figure 5-30.

5.31.1 TIME CONVERSION PROGRAM (HRSCNV)

HRSCNV converts floating-point time (seconds) into fixed-point hours, minutes and seconds when used in the various subprograms. The input parameter TIME and output parameters NHOUR, MIN, and NSEC are contained in the call statement CALL HRSCNV (TIME, NHOUR, MIN, NSEC). The output (integral hours, minutes and seconds) is placed in the parameter list of the call statement.

5.31.2 ANGLE DEGREE LIMITS DETERMINATION PROGRAM (FIXIT)

This program ensures that an angle lies between zero and 360 degrees (including zero degrees). The angle is specified in degrees in the call statement CALL FIXIT (A). If the angle does not lie between zero and 360 degrees, FIXIT modifies it by 360 degree increments until it is between the two limits. The modified angle then replaces (A), the parameter in the call statement.

5.31.3 ANGLE RADIAN LIMITS DETERMINATION PROGRAM (FIXIT1)

FIXIT1 ensures that an angle lies between zero and two pi (2π) radians (including zero radians). The angle is specified in radians in the call statement CALL FIXIT1 (A). If the angle does not lie between zero and two pi radians, FIXIT1 modifies it by two pi radian increments until it is between the two limits. The modified angle then replaces (A), the parameter in the call statement.

5.31.4 HOUR LIMITS DETERMINATION PROGRAM (FIXIT2)

FIXIT2 ensures that N hours lie between the limits of zero and 24 (including zero). The number of hours, N, is specified in the call statement CALL FIXIT2 (N). If N does not lie between zero and 24, FIXIT2 modifies it by 24-hour increments until it is between the two limits. The modified N then replaces (N), the parameter in the call statement.

5.31.5 TANGENT COMPUTATION PROGRAM (TAN)

TAN derives a function whose value is the trigonometric tangent of the parameter ANGLE. First, the angle specified in the call statement Y = TAN (ANGLE) is determined to be between zero and two pi radians (FIXIT1 ensures this). Then, TAN calculates the tangent of the corresponding first quadrant (acute) angle. The sign is determined by quadrant considerations. For infinite values, the largest possible algebraic value is set equal to the function.

5.31.6 ARC-SINE COMPUTATION PROGRAM (ARCSIN)

ARCSIN produces the radian angle between $-\pi/2$ and $+\pi/2$ whose sine corresponds to the value given in the call statement ANGLE = ARCSIN (VALUE). However, if VALUE is numerically less than + 1.0, the program uses the FORTRAN routine ATAN. If the argument exceeds + 1.0 numerically $\pm \frac{\pi}{2}$ is returned.

5.31.7 ARC-COSINE COMPUTATION PROGRAM (ARCCOS)

ARCCOS produces the radian angle between zero and π radians whose cosine corresponds to the value given in the call statement ANGLE = ARCCOS (VALUE). If VALUE is between minus and plus one (-1, +1), a transformation using ARCSIN is accomplished. If VALUE is greater than one (>1), zero is returned. If VALUE is less than or equal to minus one (\leq -1), π is returned.

5.31.8 SYSTEM TAPE WRITER (AUTHOR)

AUTHOR writes a loader on tape A1 followed by the Postflight Reporter program. The loader is written to load the Postflight Reporter program and transfer to it when the LOAD TAPE button on the console is pressed.

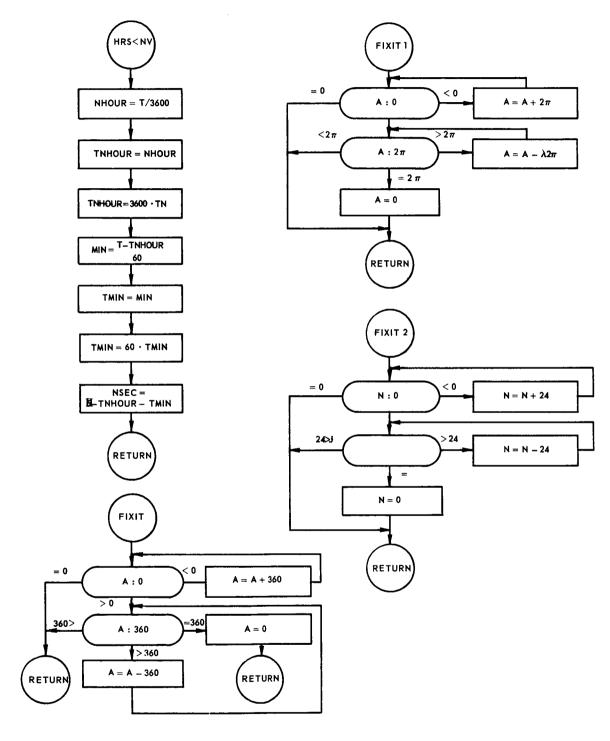


FIGURE 5-30. UTILITY PROGRAMS FLOW CHARTS (HRSCNV, FIXIT, FIXIT 1, FIXIT 2, TAN, ARCSIN, ARCCOS, AUTHOR (Sheet $1\ of\ 4$)

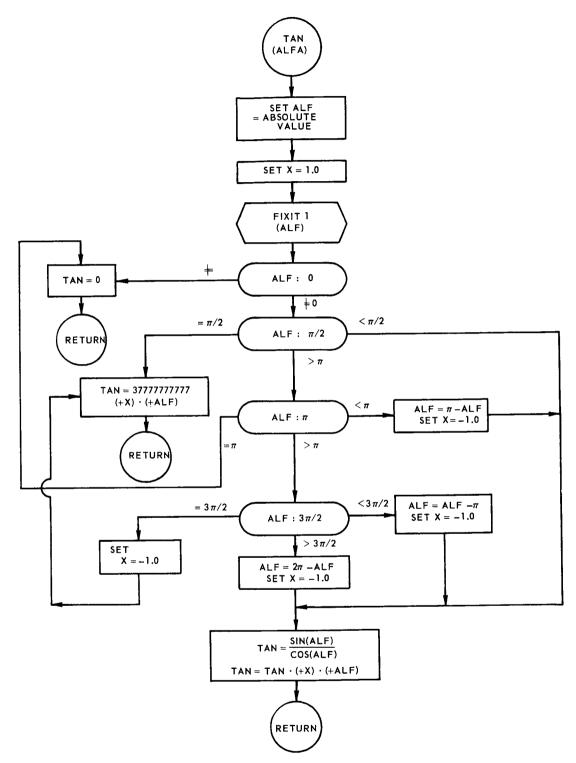


FIGURE 5-30. UTILITY PROGRAMS FLOW CHARTS (HRSCNV, FIXIT, FIXIT 1, FIXIT 2, TAN, ARCSIN, ARCCOS, AUTHOR) (Sheet 2 of 4)

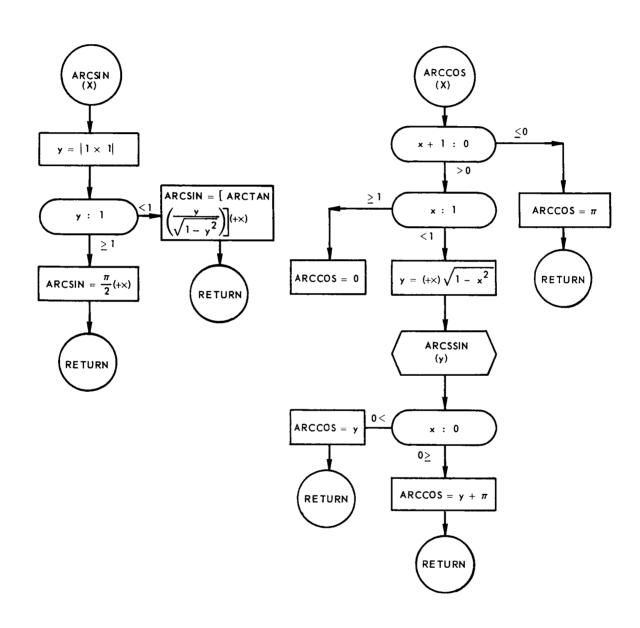


FIGURE 5-30. UTILITY PROGRAMS FLOW CHARTS (HRSCNY, FIXIT, FIXIT 1, FIXIT 2, TAN, ARCSIN ARCCOS, AUTHOR (Sheet 3 of 4)

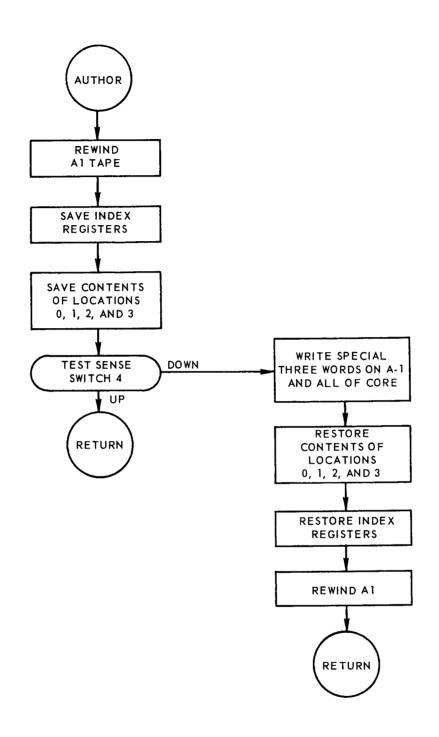


FIGURE 5-30. UTILITY PROGRAMS FLOW CHARTS (HRSCNY, FIXIT, FIXIT1, FIXIT2, TAN, ARCSIN, ARCCOS, AUTHOR) (Sheet 4 of 4)

5.32 PROGRAM OPERATING PROCEDURES

The following section lists the procedures necessary to operate the Post-flight Reporter program. The procedures are broken up into three major areas: (1) loading the program, (2) operating procedures, and (3) setting up the data deck.

5.32.1 Loading the Program

There is a choice of two methods of loading the Postflight Reporter program into the computer. The first method utilizes the 32K 7094 Fortran system. The second uses a self-loading Postflight Reporter system tape. The use of the Fortran system is the slower of the two methods. However, it offers certain advantages—it makes possible the generation of additional self-loading system tapes and it permits modifications to be made in the program.

Use of the self-loading Postflight Reporter system tape is the faster of the two methods, but it does not allow for program modification or generation of additional self-loading system tapes. Because of its advantage in speed, this is the generally preferred method since modification or generation of additional tapes is not normally needed during an operational run. The self-loading systems tape is generated by subroutine AUTHOR (see subsection 4.35.8).

5.32.2 Loading With the 32K 7094 Fortran System (Alternate No. 1)

This method is used to permit program modification and generation of additional self-loading system tapes.

- a) Tape setup.
- b) Mount the following tapes:
 - A1-32K 7094 Fortran system tape
 - A2-Postflight Reporter program deck on tape
 - A3—Blank (used by Fortran and Postflight to output computed distances)
 - A4-Blank (used by Postflight for storage of high-speed output data)
 - A5—Blank (standby to A4)
 - B1—Blank (used by Postflight as miscellaneous data tape; accepts bad data from the log tape(s) and data rejected from the programs)
 - B2—Blank (used as Condensed Postflight Report output tape)
 - B3—Blank (used by Fortran and Postflight as an output plot tape)

- B4—Blank (used by Postflight for storage of high-speed input messages; first file for high-speed input data; second file for a record of the seven discrete events for GETME)
- B6-Mercury Programming System log tape
- B7-Standby to B6
- C3-Blank (used as output tape for the Postflight Report)
- c) Place a 32K Fortran start card in the card reader.
- d) Clear the computer and press LOAD CARD(S).

After the program has been loaded, it will print the following message and halt with a 525258 in the address of the instruction register: IF A SYSTEM TAPE IS TO BE GENERATED - MOUNT A BLANK A1, DEPRESS SENSE SWITCH 4 AND START. IF NOT, MERELY PRESS START.

These instructions should then be followed and a self-loading Postflight Reporter system tape, either generated or not. This completes the loading phase using the Fortran system.

5.32.3 Loading With the Self-loading Postflight Reporter Program System Tape (Alternate No. 2)

This method is used when program modifications are not needed or additional system tapes are not to be generated.

- a) Tape Setup.
- b) Mount the following tapes:
 - A1-Self-loading Postflight Reporter program system tape
 - A3—Blank (used by Postflight to output computed distances)
 - A4—Blank (used by Postflight for storage of high-speed output data)
 - A5—Blank (standby to A4)
 - B1—Blank (used by Postflight as miscellaneous data tape; accepts bad data from log tape, and data rejected from the programs)
 - B2—Blank (used as Condensed Postflight Report output tape)
 - B3-Blank (used as output plot tape)
 - B4—Blank (used by Postflight as high-speed input message tape; first file for high-speed input data; second file for record of the seven discrete events for GETME)

B6—Mercury Programming System Log Tape

B7-Standby to B6

C3—Blank (used as output tape for the Postflight Report)

c) Clear the computer and press LOAD TAPE.

The program will load itself into the computer and halt, completing the loading phase.

5.33 OPERATING PROCEDURES

This section lists the procedures necessary for actual operation of the Postflight Reporter program once it has been loaded into the computer.

Upon completion of the loading of the program, the computer prints the following operating instructions and halts with 777718 in the address of the instruction register: IF SORT IS NECESSARY - ENTER THE NUMBER OF PHYSICAL LOG TAPES IN THE ADDRESS OF THE KEYS AND PRESS START. IF SORT IS TO BE SKIPPED - DEPRESS SENSE SWITCH 6, MOUNT A4 AND B4, AND PRESS START. SENSE SWITCH 1 MAY BE TOGGLED FOR ON-LINE GLIMPSES OF C3 OUTPUT. DEPRESS SENSE SWITCH 3 TO SUPPRESS CONDENSED REPORT ON B2. DEPRESS SENSE SWITCH 2 TO SUPPRESS DISTANCE COMPUTATION.

The settings of all switches and keys, and the insertion of the data deck are performed at this time. The operation is as follows:

a) Entry Keys

The number of physical log tapes is right-justified in the address portion of the entry keys.

b) On-line Card Reader

Data deck in and ready (see subsection 5.36.3 for setup of this deck).

c) Sense Switch Settings

SS1 Up—suppress on-line copy of C3

Down-print copy of C3 on-line

SS2 Up-compute and write distances on A3

Down—suppress computation of distances

SS3 Up-write Condensed Report on B2

Down—suppress writing of Condensed Report on B2

SS6 Up—execute SORTER (use Mercury Programming System Log Tapes as input to prepare A4 and B4 data tapes)

Down—suppress SORTER (use prepared A4 and B4 data tapes from a previous run)

After all settings have been made, the start button is pressed and operation begins. During operation of the programs, certain error conditions may be determined. When this occurs, the program prints a message giving the condition existing and halts, awaiting action by the operator.

d) Program Stops

The program stops and the associated messages are as follows:

- HPR 77773₈—error stop from GETME, on-line printout, AN ERROR HAS BEEN DETECTED IN KEY OR TAPE SET-UP. PLEASE REVIEW OPERATING NOTES AFTER COMPLETING CORRECTION PRESS START.
- HPR 77775 $_8$ -error stop from SORTER, same on-line printout as for 77773 $_8$.
- HPR 77777₈—end of job has been reached. EOF has been written on C3, A3, B3, and B2. Print C3, A3, and B2 under program control at six lines per inch. To run additional cases, mount new input tape and blanks on B2 and C3. Press start.
- HPR 12121₈—error stop for request-for-request card, on-line printout, ERROR IN DECK SET-UP. TF EXCEEDS TL. REPUNCH, PRESS START.

5.34 SETUP OF DATA DECK

This subsection describes the preparation and setup of the data deck. Six types of cards are in the deck, the format of which is listed below.

	1	T	1
Card Type	Card Columns	Digital Formats	Data
1	1-72 (inclusive)	72 Alphanumeric Characters	Headings printed at beginning of the output
2	1-11	xxx.xxxxxx	Greenwich hour angle at midnight GMT preceding launch (XNUA) degrees
	12-22	xxx,xxxxxx	Launch azimuth (THETAO) degrees
	23-27	xxxxx	Computer name (A or B)
3	1	x	1 = Three-Day Report; 2 = Quick Look
	2-3	xx	Integral hours/minutes/seconds =
	4–5	xx	TF = first time since phase
	6-9	xx.x	initiation
	10-11	xx	Integral hours/minutes/seconds =
	12-13	xx	TL = last elapsed time since phase
	14-17	xx.x	initiation
	18-22	xxx.x	Time interval (Δt) between report output records, seconds
	23	x	Phase indicator 1 = Launch, 2 = Abort, 3 = Orbit, 4 = Reentry
4	1	x	0 = last vector in phase 1 = not last vector in phase
	2-8	xxxxx.x	Anchor time of vector in GMT seconds
	9–15	XXXXX.X	Last time for which vector is valid in GMT seconds

Setup of Data Deck (Continued)

MC 63-4

Card Type	Card Columns	Digital Formats	Data
	16-22	xxxxx.	Time since anchor time (Δt) in seconds at which to start using vector.
5	1-12	xxxxxxxxxx	$\overline{\underline{X}}$ component of inertial position vector in floating-point octal
	13-24	xxxxxxxxxx	$\frac{\overline{Y}}{}$ component of inertial position vector in floating-point octal
	25-36	xxxxxxxxxx	$\overline{\underline{Z}}$ component of inertial position vector in floating-point octal
	37-48	xxxxxxxxxx	$\frac{\dot{\overline{X}}}{\dot{\overline{X}}}$ component of inertial velocity vector in floating-point octal
	49-60	xxxxxxxxxx	$\frac{\dot{\overline{Y}}}{Y}$ component of inertial velocity vector in floating-point octal
	61-72	xxxxxxxxxx	$\frac{\dot{\overline{Z}}}{\overline{Z}}$ component of inertial velocity vector in floating-point octal
6	1	х	Sign-off. Zero punch in Column 1 to indicate end of run

The data deck is setup in the following manner. The first two cards of the deck are always of types 1 and 2 in that order (see Figure 5-31. Type 1 is a heading card which is printed with the output. Type 2 is a physical parameter card describing the launch day. These cards are then followed by a series of card groups containing one or more cards for each flight phase desired in the output.

The launch and abort phases each have one card—a phase card of type 3. The orbit and reentry phases each have a phase card of type 3 followed by a pair of time, position and velocity cards (types 4 and 5) for each input vector. The last card of the data deck is of type 6 which signals the end of the run.

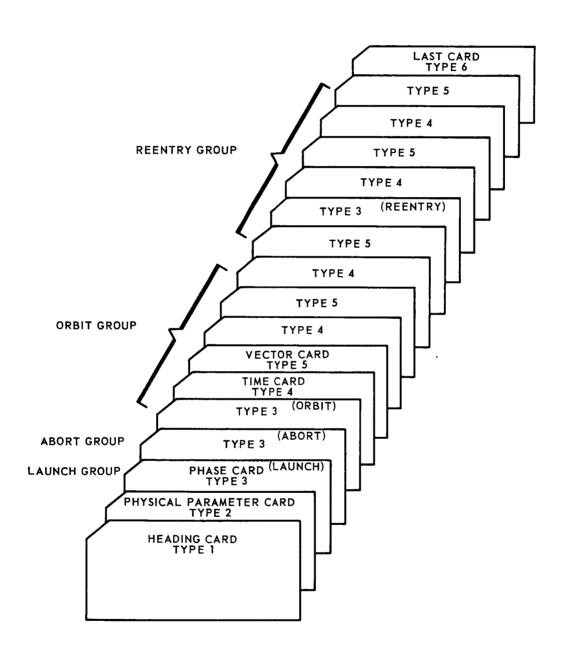


FIGURE 5-31. DATA DECK SETUP

Appendix A

INDEX TO PROGRAMS

Program	Manual	Sec.	Title
A3MSCP	MC63 3	2	Compute Latitude and Longitude Program
ACTORS	MC63 4	5	Constant Factors Initialization Program
AJACQ	MC63 2	6	Acquisition Data Macro
ARCCOS	MC63 4	5	Arc-Cosine Computation Program
ARCSIN	MC63 4	5	Arc-Sine Computation Program
A0STAD	MC63 3	3	Acquisition Data Generation Program
ATMOS	MC63 4	5	Atmospheric Density Processor Program
AUTHOR	MC63 4	5	System Tape Writer
B2SWIP	MC63 3	2	Sliding Wire Impact Predictor Program
BCD	MC63 3	4	Binary Coded Decimal Conversion Program
BCLS2	MC63 4	2	Closed Loop Simulation Program, Bda to Goddard
BCTB	MC63 4	5	BCD Word Conversion Program A (BCTB1)
BINNO	MC63 2	7	Binary Number Card Image Macro
BOSCER	MC63 3	2	Earth Radius Determination Program
BYERR	MC63 3	2	Launch CHNOND Error Return Processing Section
C9ASKE	MC63 4	1	Solution of Kepler's Equation Subroutine
C9DTRF	MC63 3	2	Retrofire Time Correction Program
C9DTRF	MC63 3	3	Retrofire Time Correction Program
C9RVTH	MC63 4	1	Computation of Elliptic Motion During Launch Subrtn
CALTOI	MC63 3	2	Short Arc Subrtn to Convt Loc RAE to Inert XYZ
CAMPSO	MC63 3	2	Short Arc Subrtn to Approx Mid-Point Solution
CANORM	MC63 3	2	Short Arc Subrtn to Form Norm Equation Sums

Program	Manual	Sec.	Title
CASARC	MC63 3	2	Goddard Short Arc Orbit Determination Program
CASOLN	MC63 3	2	Short Arc Subroutine to Solve Normal Equations
CC7091	MC63 3	2	IP 7094 Coordinate Transformation Program
CCABRT	MC63 3	2	High Abort Program
CCGEB1	MC63 3	2	B-GE Data Conversion Program
CCHOLD	MC63 3	2	Hold Phase Main Calculation Program
CCHOMI	MC63 3	2	Missing Data During Hold Phase Routine
CCLOK	MC63 3	2	Launch Prog to Insure Retrofire Occurs in Bda Comm. Capability
CCMAIN	MC63 3	2	Launch Phase Main Calculation Program
CCMEAB	MC63 3	2	Medium Abort Program
CCMISS	MC63 3	2	Missing Data During Normal Launch Routine
CCMLR	MC63 3	2	Launch GMTRC Computation Routine
CCMLS	MC63 3	2	Launch Init and Sel Source Data Coll Sect
CCMLT	MC63 3	2	Launch Pre-Tower Separation Logic
CCMLV	MC63 3	2	Launch Logic Providing Vectors to Make Go- No-Go Decision
CCMLX	MC63 3	2	Launch Average Vector Data Switching Routine
CCRTYL	MC63 3	2	Low Abort Program (CCRTMI)
CCSTIP	MC63 3	2	Strip Chart Processor Programs (CCSTGE)
CFDTR	MC63 3	2	Launch Phase Output Adjustment Program
CHNOND	MC63 3	2	Computation of Circular Orbit Program
CHUMLY	MC63 4	5	BCD Output Initialization Program
CKEN	MC63 3	2	Launch Phase Time Conversion Subroutine
CKIC	MC63 3	2	Launch Phase Inclination Angle Computation Routine
CKNOC	MC63 3	2	Launch Phase Impact Prediction Routine 5
CKNOD	MC63 3	2	Launch Phase Impact Predictions Logic for Velocities of 10K FPS when Perigee is $<430\mbox{K}$ Ft
CKYS	MC63 3	2	Launch Ph IP Logic for Velocities ≥ 10K FPS when Perigee ≥ 430K Ft

Program	Manual	Sec.	Title
CLS3	MC63 4	2	Closed Loop Simulation Program MCC
CMLJ	MC63 3	2	Launch Go-No-Go Decision Logic
COL8ER	MC63 4	3	Symbolic Tape Updating Program
COMPAR	MC63 4	3	Squoze Deck Comparison Program
CORING	MC63 4	4	Prog to Init Taking Snap Dumps of the Mer Sys
CORMAP	MC63 4	3	Core Mapping Program
CR3R4	MC63 3	2	Launch Phase Impact Prediction Routine 3
CR3R6	MC63 3	2	Launch Phase Impact Prediction Routine 2
CR5HE	MC63 3	2	Launch Phase Impact Prediction Routine 4
CR5R6	MC63 3	2	Launch Phase Impact Prediction Routine 1
CSTLU	MC63 3	2	Launch Ph Impact Pred Logic for Vect < 10K FPS
D1PCDC	MC63 3	3	Partial Coefficient Calculation Program
D2CSFE	MC63 3	3	Calculate Standard Error Fit Program
D2SLNE	MC63 3	3	Solve Normal Equations Program
D2SSMT	MC63 3	3	Station Sum of Squares Over M Program
D2SUNE	MC63 3	3	Set Up Normal Equations Program
D2WEQT	MC63 3	3	Weighting the Equations of Condition Program
D3CDRM	MC63 3	3	Choose Desired Radar Message Program
D3PTWT	MC63 3	3	Weight Table Lookup Program
D4CFRV	MC63 3	3	Calculate Functions of R and V Program
DGHR	MC63 3	4	Time Measure to BCD Degrees Conversion Program
DISTAN	MC63 4	5	Distance Computation of Earth and Space Track
DONIN	MC63 4	5	High-Speed Input Processor Program
DONOUT	MC63 4	5	High-Speed Output Processor Program
DODIFC	MC63 3	3	Differential Correction Control Program
E0BDIT	MC63 3	2	Bermuda High-Speed Edit Program
E0LEDI	MC63 3	3	Editing Program
FIXIT	MC63 4	5	Angle Degree Limits Determination Program

Program	Manual	$\frac{\text{Sec.}}{}$	Title
FIXIT1	MC63 4	5	Angle Radian Limits Determination Program
FIXIT2	MC63 4	5	Hour Limits Determination Program
GCNVE	MC63 4	5	Units Conversion Program
GEB	MC63 4	5	Subchannel 1 Processing Program
GECNV	MC63 4	5	B-GE Reference Frame Conversion Program
GETME	MC63 4	5	Discrete Event Processor Program
GRUNGY	MC63 4	5	Langrangian Interpolation
HBSUB	MC63 4	2	HB Subroutine
HCSUB	MC63 4	2	HC Subroutine
HEAT	MC63 4	5	Stagnation Heat Rate Processor Program
HMSTS	MC63 4	5	Time Word Conversion Program (B)
HOMER	MC63 4	4	Program to Write Dumping Portion of B4 Tape
HRSCNV	MC63 4	5	Time Conversion Program
HSIN7	MC63 4	3	Log Tape High-Speed Input Program
HSOP1	MC63 4	5	High-Speed Output Tape Writer Program
IBWR1	MC63 4	1	Share System Tape Writer Program
IBWR2	MC63 4	1	Share System Tape Editor Program
INITIA	MC63 4	5	System Parameter Initialization Program
IPCNV	MC63 4	5	IP 7094 Reference Frame Conversion Program
IPORR	MC63 4	5	IP 7094 High-Speed Input Processor Program
I0HSBD	MC63 3	1	Bermuda High-Speed Input Program
I0HSGB	MC63 3	1	B-GE High-Speed Input Program
I0HS09	MC63 3	1	IP-7094 High-Speed Input Program
I0MANI	MC63 3	1	Manual Insertion Program
IOTTIN	MC63 3	1	Low-Speed Teletype Input Program
ISODMP	MC63 4	4	Isolated Dumping Portion of B4 Tape
KEYCF	MC63 2	2	Mon Subrtn to Ind CADFISS Use of Radar Stations
KEYDC	MC63 2	2	Mon Subrtn to CTL for Insertions and Deletions in TMSTMS Tbl
KEYDE	MC63 2	2	Mon Subrtn to Bypass Diff Corr for a Station

Program	Manual	Sec.	Title
KEYRT	MC63 2	2	Monitor Subroutine to Accept GMT of Retrofire
KEYS	MC63 4	3	Tape Key Comparison Program
KEYTL	MC63 2	2	Monitor Subroutine to Insert or Delete Telemetry
LAUNCH	MC63 4	5	Launch Phase Processor Program
LBRWR	MC63 4	1	SOS Library Tape Writer Program
LOWCOR	MC63 4	3	Low Core Reference Program
MANIN	MC63 4	5	Manual Insertion Processor Program
MBCD	MC63 3	4	BCD to Modified BCD Conversion Program
MERCNV	MC63 4	5	True Inertial Coordinate Conversion Program
MERGE	MC63 4	2	Merge Program
MFABRT	MC63 2	3	Monitor Suffix to CCABRT
MFCCGB	MC63 2	3	Monitor Suffix to CCGEB1
MFCCGE	MC63 2	3	Monitor Bad Data Suffix to CCGEB1
MFCCIP	MC63 2	3	Monitor Suffix to CC7091
MFCPNI	MC63 2	5	Monitor Suffix to NOCPNI
MFDIFC	MC63 2	5	Monitor Prefix to D0DIFC
MFE0BD	MC 63 2	3	Monitor Suffix to Bermuda Edit Processor E0BDIT
MFHSBD	MC63 2	2	Monitor Suffix to I0HSBD
MFHSGB	MC63 2	2	Monitor Radar Data Suffix to I0HSGB
MFHS08	MC63 2	2	Monitor Telemetry Suffix to I0HS09
MFHS09	MC63 2	2	Monitor Radar Data Suffix to I0HS09
MFLABT	MC63 2	3	Monitor Abort Suffix to MFLHLD
MFLANA	MC63 2	6	Monitor Suffix to O0LANA
MFLED1	MC63 2	5	Monitor Suffix to E0LED1
MFLHLD	MC63 2	3	Monitor Suffix to CCHOLD and CCHOMU
MFLNML	MC63 2	3	Monitor Suffix to CCMAIN and CCMISS
MFLPBD	MC63 2	6	Monitor Suffix to O0LPBD
MFLRT1	MC63 2	3	Monitor Suffix to CCRTYL and CCRTMI
MFLRT2	MC63 2	3	Monitor Suffix to CCMEAB

Program	Manual	Sec.	Title
MFMAN1	MC63 2	2	Monitor Time of Liftoff Suffix to IOMANI
MFMAN2	MC63 2	2	Monitor Normal Retrofire Suffix to IOMANI
MFMAN3	MC63 2	2	Monitor Long and Pass No. Suffix to IOMANI
MFMAN4	MC63 2	2	Monitor Delta T and Reentry Time Suf to IOMANI
MFMAN5	MC63 2	2	Monitor R and V Suffix to IOMANI
MFMAN6	MC63 2	2	Monitor Capsule Clock Setting to IOMANI
MFMAN7	MC63 2	2	Monitor Non Nominal Retrofire Suffix to IOMANI
MFMAN8	MC63 2	2	Monitor Orbit Constants to IOMANI
MFMAN9	MC63 2	2	Monitor DC Intvl and Arc Lgth Suffix to I0MANI
MFMANI	MC63 2	2	Monitor Suffix to IOMANI
MFMANR	MC63 2	2	Monitor General IOMANI Suffix Exit
MFMAOS	MC63 2	2	Monitor Abort/Orbit Switch Suffix to IOMANI
MFML6A	MC63 2	2	Monitor Telemetry Suffix to IOHSGB
MFORMC	MC63 2	5	Monitor Suffix to O5ORMC
MFORRE	MC63 2	6	Monitor Suffix to OOORRE
MFRARF	MC63 2	5	Monitor Suffix to R5RARF
MFSARC	MC63 2	3	Monitor Suffix to Short Arc Program
MFSTRP	MC63 2	3	Monitor Suffix to the Strip Chart Processor
MFTTIN	MC63 2	2	Monitor Suffix to IOTTIN
MLEXBD	MC63 2	3	Monitor Launch Bermuda Vector Extrap Sub- routine
MLSWCH	MC63 2	3	Monitor Launch Table Switching Subroutine
MLUPDT	MC63 2	3	Monitor Launch Output Updating Subroutine
MMAQA	MC 63 3	2	Launch Phase Data Acquisition Program
MPABRT	MC63 2	3	Monitor Prefix to CCABRT
MPCCGB	MC63 2	3	Monitor Prefix to CCGEB2
MPCCIP	MC63 2	3	Monitor Prefix to CC7091
MPCPNI	MC63 2	5	Monitor Prefix to NOCPNI
MPDIFC	MC63 2	5	Monitor Prefix to D0DIFC

Program	Manual	Sec.	Title
MPDIFK	MC63 2	5	Monitor Prefix to D0DIFC
MPDIFM	MC63 2	5	Monitor Prefix to D0DIFC
MPEOBD	MC63 2	3	Monitor Prefix to Bermuda Edit Processor E0BDIT
MPE0BD	MC63 2	3	Monitor Prefix to Bermuda Edit Processor E0BDIT
MPHSBD	MC63 2	2	Monitor Prefix to IOHSBD
MPHSGB	MC63 2	2	Monitor Prefix to IOHSGB
MPHS09	MC63 2	2	Monitor Prefix to I0HS09
MPLANA	MC63 2	6	Monitor Prefix to OOLANA
MPLCCM	MC63 2	3	Monitor Prefix to Main Launch Computations
MPLED1	MC63 2	5	Monitor Prefix to E0LED1
MPLED2	MC63 2	5	Monitor Prefix to E0LED1
MPLPBD	MC63 2	6	Monitor Prefix to O0LPBD
MPORMC	MC63 2	5	Monitor Prefix to O5ORMC
MPORRE	MC63 2	6	Monitor Prefix to OOORRE
MPRARF	MC63 2	5	Monitor Prefix to R5RARF
MPSARC	MC63 2	3	Monitor Prefix to Short Arc Program CASARC
MPSTRP	MC63 2	3	Monitor Prefix to the Strip Chart Processor
MPTTIN	MC63 2	2	Monitor Prefix to IOTTIN
M0DIAG	MC63 2	1	Main Controller Diagnostic Program
M0ENDS	MC63 2	1	Main Ending Program
MOINIT	MC63 2	4	Main Controller Initialization Program
MOPANL	MC63 2	1	Non Real Time Save Program
M0PRIO	MC63 2	1	Main Controller Priority Program
M0QUEU	MC63 2	1	Main Controller Queue Program
M0RTCC	MC63 2	1	Real Time Channel Main Controller Program
MORTRN	MC63 2	1	Main Controller Return Program
MOSAVE	MC63 2	1	Main Controller Save Program
MOUNQU	MC63 2	1	Main Controller Unqueu Program

Program	Manual	Sec.	$\underline{\underline{\text{Title}}}$
MSDELT	MC63 2	5	Mon Subrtn to Delete Entries in TMSTMS Table
MSDEOV	MC63 2	2	Monitor Overlay Sort Subroutine
MSIINT	MC63 2	5	Subrtn to Remove Obs from Differential Correction
MSINSR	MC63 2	5	Subrtn to Insert Radar Obs into Diff Correction
MSLCUC	MC63 2	5	Mon Subrtn to Move Radar Msg Blk to 32K Buffer
MSLOGG	MC63 2	7	Monitor Logging Subroutine
MSMOVE	MC63 2	7	Monitor Subroutine for Moving Data Between Cores
MSPASN	MC63 2	2	Monitor Pass Number Subroutine
MSRECC	MC63 2	4	Error Correction Code Reader Subroutine
MSTIAC	MC63 2	2	Monitor Teletype Input Acceptability Subroutine
MSTICK	MC63 2	2	Monitor Teletype Input Check Subroutine
MSVTCA	MC63 2	5	Generation of TMVTCA Table
MSWECC	MC63 2	4	Error Correction Code Writer Subroutine
MSWTMS	MC63 2	5	Mon Subrtn to Comp a TMDATA Block Address
MTCPCC	MC63 2	2	Monitor Prog Ctl Console Channel Trap Processor
MTDDCP	MC63 2	6	Monitor Trap Processor for O0LPBD
MTENLG	MC63 2	7	Monitor End-of-Logging Trap Processor
MTENPR	MC63 2	7	Monitor End-of-Printing Trap Processor
MTENST	MC63 2	4	Monitor Station Characteristics Tape Processor
MTERTC	MC63 2	2	Monitor Real Time Channel Error Trap Processor
MTHFSC	MC63 2	2	Monitor Half Second Trap Processor
MTHSB5	MC63 2	2	Monitor Bermuda HS Input Trap Proc for SC5
MTHSB6	MC63 2	2	Monitor Bermuda HS Input Trap Proc for SC6
MTHSGB	MC63 2	2	Monitor B-GE High Speed Input Trap Processor
MTHSOD	MC63 2	6	Mon Trap Proc for HS Output To CC

Program	Manual	Sec.	Title
MTHSOP	MC63 2	6	Mon Trap Proc for HS Output to Goddard
MTHS09	MC63 2	2	Monitor IP 7094 High Speed Input Trap Processor
MTINTV	MC63 2	4	Monitor Interval Timer Trap Processor
MTMSCK	MC63 2	7	Monitor Message Check Trap Processor
MTQTSX	MC63 2	1	Simulated Trap Processor
MTRRRS	MC63 2	4	Monitor Reentry Restart Tape Trap Processor
MTRSTC	MC63 2	4	Monitor Orbit Restart Tape Trap Proces (MTRSTB)
MTRSYS	MC63 2	4	Monitor System Tape Processor
MTSENS	MC63 2	6	Monitor Sense Output Trap Processor
MTTEST	MC63 4	4	Real Time Transfer Trapping Test Program
MTTTIN	MC63 2	2	Monitor Teletype Trap Processor
MTTTOX	MC63 2	6	Monitor Teletype Output Trap Processors (MTTTOY)
MTWRRS	MC63 2	4	Monitor Restart Tape Writer Trap Proces (MTWRS1)
MTWWVI	MC63 2	2	Monitor Initial WWV Trap Processor
MTWWWV	MC63 2	2	Monitor WWV Trap Processor
MXCHER	MC63 4	3	Program Providing Selection DCC Subchnl IO Data from Mercury Log Tape
MXCORE	MC63 4	3	Symbolic Core Dump Program
MXDEFN	MC63 4	4	Extended Definition of Symbols Processor
MXHSPL	MC63 4	3	Log Tape Plotting Program
MXHSPR	MC63 4	3	Log Tape High-Speed Output Program
MXILCO	MC63 4	3	Program to Print Real Time Core'd Output
MXLOAD	MC63 4	4	Mercury System Tape Loader
MXMRGE	MC63 4	4	Monitor Merge Program
MXNDKT	MC63 4	3	Priority Indicator Listing Program
MXPOCL	MC63 4	3	Program to Print Mercury Log Tape in Octal
MXPRLG	MC63 4	3	Log Tape Printer Program

Program	Manual	Sec.	Title
MXSTW1	MC63 4	4	Mercury System Tape Writer Program
MXTHLG	MC63 4	3	Low-Speed Output Printer Program
MXWMOT	MC63 4	4	Message Tape Writer Program
MYACQD	MC63 2	6	Monitor Acquisition Data Processor
MYBUFR	MC63 2	7	Monitor Buffering Processor
MYDCIT	MC63 2	4	Mon Edit Processor for Orbit to Reentry Interface
MYDDCP	MC63 2	6	Monitor Output Processor for O0LPBD
MYGEN1	MC63 2	5	Monitor Numerical Integration Generator
MYGEN2	MC63 2	5	Monitor Numerical Integration Generator
MYGEN3	MC63 2	5	Monitor Numerical Integration Generator
MYHSOD	MC63 2	6	Mon HS Output to Cape Canaveral Processor
MYHSOP	MC63 2	6	Mon HS Output to Goddard Processor
MYINIT	MC63 2	4	Monitor Initialization Processor
MYKEYS	MC63 2	2	Monitor Processor to Use PCC Entry Switches
MYLSYS	MC63 2	4	Monitor Systems Tape Loading Processor
MYMESS	MC63 2	7	Monitor On-Line Message Processor
MYMINS	MC63 2	2	Monitor Minute Processor
MYMSCK	MC63 2	7	Monitor Message Check Processor
MYQNEA	MC63 2	5	Monitor Queue Next Emergency Area Processor
MYQPRA	MC63 2	5	Monitor Queue Primary Area Processor
MYQSYS	MC63 2	4	Monitor System Tape Queueing Processor
MYREST	MC63 2	4	Mon Orbit to Reentry Interphase Processor
MYRRRS	MC63 2	4	Monitor Reentry Restart Tape Processor
MYRSYS	MC63 2	4	Monitor System Tape Trap Processor
MY0PCC	MC63 2	2	Monitor Program Control Console Processor
MYSCRD	MC63 2	4	Monitor Station Characteristics Tape Processor
MYSEEK	MC63 2	3	Monitor High Abort Control Processor
MYSENS	MC63 2	6	Monitor Sense Output Processor
MYSRST	MC63 2	4	Monitor Orbit Restart Tape Processor

APPENDIX B

POSTFLIGHT REPORTER SYMBOLIC DESIGNATIONS

This appendix includes both the FORTRAN and the mathematical symbology used to produce the postflight reports. The material in the FORTRAN list is arranged alphabetically. Coordinate systems appearing at the end of this appendix are defined under the appropriate heading in Appendix B, with the exception of the last one—the intermediate rotational coordinate system which is covered in NASA Working Paper 146.

FORTRAN Symbolic Names	Mathematical Symbology	Definition
A	a s	Spherical radius of earth
AC	a C	Equatorial radius, Clarke 1866 Spheroid
ARBAR	a-R	Semi-major axis less spherical radius
AREA	Area	Selected recovery area
ARGP	ω	Argument of perigee (defined in volume MC 102)
ARGP1	ώ	Angular rotation of ω
ASUBR	^A R	Resultant acceleration—radial component of the instantaneous acceleration, g units
AXIS	a	Semi-major axis
ВС	$^{\mathrm{b}}\mathrm{_{c}}$	Polar radius, Clarke 1866 Spheriod
CANJ2	$^{\mathrm{J}}_{\mathrm{2}}$	Canonical 2nd harmonic potential
CANJ3	J_3	Canonical 3rd harmonic potential
CANJ4	J_4	Canonical 4th harmonic potential
CL	L _C	Geocentric latitude

FORTRAN Symbolic Names	Mathematical Symbology	Definition
CR	y-y nominal	Crossrange distance
CS	c_{s}	Speed of sound
D	d	Downrange distance
DENS	ρ	Atmospheric density
DL	L_{D}	Geodetic latitude—latitude of present position
DLMI	Δλ _i	Angle in equatorical plane between X and the projection of capsule into that plane
DPHIR	$\delta heta_{ m R}$	Angle at 2° lift-off in the equatorial plane between T and longitude of GE radar
DTR	$\Delta t_{f r}$	Time delay to retrofire
DTHTAP	$\delta heta_{ m p}$	Angle in equatorial plane between T and longitude of pad
EA	E	Eccentric anomaly, degrees
ECC	e	Eccentricity (defined in Volume MC 102)
ECTRC	ECTRC	Elapsed capsule clock time of retro- fire computed
ECTRC1	ECTRC ₁	ECTRC for emergency recovery area
ECTRC2	\mathtt{ECTRC}_2	ECTRC for end of present orbit
ECTRC3	ECTRC ₃	ECTRC for end of normal 3-orbit mission
ECTRS	ECTRS	Elapsed capsule time to retrofire, presently set
EGT	EGT	Elapsed ground time since retrofire occurred

FORTRAN Symbolic Names	Mathematical Symbology	Definition
EQURAD	$^{\mathrm{a}}\mathrm{e}$	Canonical equatorial radius
FFLAT	f	Canonical flattening
GE	γ _e	Earth-fixed flight-path angle
GI	γ _i	Inertial flight-path angle (positive above local horizon), degrees
GIGE	$(\gamma_i - \gamma_{nom})GE$	Actual minus nominal flight-path angle for B-GE data, degrees
GIIP	$(\gamma_i - \gamma_{nom})$ IP	For IP-7090 data actual minus nominal flight-path angle for IP 7090 data, degrees
GMTLC	GMTLC	Greenwich mean time of landing
GMTLO	GMTLO	Greenwich mean time of 2" lift-off
GMTRC	GMTRC	Greenwich mean time of retrofire computed
GMTRC1	\mathtt{GMTRC}_1	GMTRC for emergency recovery area
GMTRC2	\mathtt{GMTRC}_2	GMTRC for end of present orbit
GMTRC3	GMTRC ₃	GMTRC for end of normal 3-orbit mission
GMTRS	GMTRS	Present capsule retrofire clock setting computed
GRAVIT	ġ e	Canonical value of gravity at equator
GTL	GTL	Time until landing
GTRS	GTRS	Time left until retrofire will occur
НА	h _a	Apogee height
HE	$^{ m h}{_{ m e}}$	Height above oblate earth
HPAD	^h o	Height of pad above mean sea-level

FORTRAN Symbolic Names	Mathematical Symbology	Definition
OME	Ω	Longitude of ascending node (defined in volume MC 102)
OME1	$\dot{\Omega}$	Angular rotation of Ω
OMEGAE	$\omega_{ m e}$	Angular rotation of earth
P	p	Semi-latus rectum
PER	T	Orbital period, minutes
РНІ	φ	Total angle in plane from longitude of node to capsule
PHIIP	$oldsymbol{\phi}_{ ext{IP}}$	Geodetic latitude of refired impact point
PHIMAX	$\phi_{ ext{max}}$	Geodetic latitude - maximum time delay - impact point, degrees
PHIMIN	ϕ_{\min}	Geodetic latitude - minimum time delay - impact point, degrees
PSIE	$m{\psi}_{ ext{e}}$	Earth-fixed heading
PSII	ψ_{i}	Inertial heading
QD	q^{D}	Dynamic pressure
QS	$\mathbf{q}_{\mathbf{s}}$	Heating rate, BTU/ft ² second
RADIUS	r	Geocentric radial distance to the capsule, feet
RBAR	\overline{R}	Radial distance of a spherical earth
RL	$^{\mathbf{r}}_{1}$	Local radius of earth (i.e., at the geocentric latitude of capsule)
RN	R_{N}	Reynolds number
RPBAR	$ar{\mathtt{R}}_{\mathtt{p}}$	Geocentric radius at pad
RRBAR	r - R	Height above spherical earth, nautical miles, $\overline{R} = 20,910,000$ feet

FORTRAN Symbolic Names	Mathematical Symbology	Definition
R01	ro	$(=\overline{R}_p + h_o =)$ Total distance from geocenter to pad
S	S	Range from launch pad
SR	${ m s}_{ m R}$	Re-entry range from retrofire, nautical miles
T	t	Elapsed time since lift-off
TA	θ_{1}	True anomaly
THETA0	θ_{0}	Launch azimuth, from north, positive clockwise
TP	$\mathbf{t}_{\mathbf{p}}$	Elapsed time from perigee passage
TRETRO	$\mathbf{t_r^P}$	Time of retrofire
VE	\overline{v}_e	Earth-fixed velocity, ft/sec.
VI	$\mathbf{v_i}$	Inertial velocity, ft/sec.
VIVR	v_i/v_R	Speed ratio
VIVRGE	$(v_i/v_R^{-1}v_i/v_R^{nom})GE$	Actual minus nominal velocity ratio for B-GE data
VIVRIP	$(v_i/v_R^{-}v_i/v_R^{nom})$ IP	Actual minus nominal velocity rate for IP 7090 data
XICTRC	ICTRC	Incremental change to reset clock
XINC	i	Inclination angle (defined in volume MC 102)
XL	λ	Earth-fixed longitude
XLAMP	$\lambda_{ m p}$	Longitude of perigee at perigee passage
XLMAX	$\phi_{ m max}$	Longitude - maximum time delay - impact point, degrees
XLM1	λ_{1}	Longitude of launch pad

FORTRAN Symbolic Names	Mathematical Symbology	<u>Definition</u>				
XLMMIN	λ _{min}	Longitude - minimum time delay - impact point, degrees				
XLMIP	$\lambda_{ m IP}$	Longitude of refired impact point				
XLPAD	$\mathtt{L}_{\mathbf{P}^{'}}$	Geodetic latitude of pad				
XLRHO	$^{ extsf{L}_{oldsymbol{ ho}}}$	Geocentric latitude of pad				
XMA	M_{a}	Mean anomaly				
XMACH	M	Mach number				
XMM	$oldsymbol{\eta}_{ ext{M}}$	Mean motion				
XMUE	$\mu_{\mathbf{e}}$	Earth's gravitational constant (0 th order harmonic)				
NORBCP	NORBCP	Orbit capability				
XNU	ν	Kinematic viscosity				
XNUA	$\nu_{\rm a}$	Greenwich hour angle from T at midnight preceding launch				
NUMORB	NUMORB	Orbit number				
X	$\overline{\mathbf{X}}$					
Y						
Z	<u>Y</u>	True Inertial coordinate system				
X1	$egin{array}{c} \dot{ar{\mathbf{x}}} \ \dot{ar{\mathbf{y}}} \ \dot{ar{\mathbf{z}}} \end{array} ight]$	(See Appendix B)				
Y 1	$\dot{\overline{\mathbf{Y}}}$					
Z1	<u>Ż</u>					
U	u)					
V	v					
W	w	Pad coordinate system				
U1	ů	(See Appendix B)				
V1	v					
W1	w)					

FORTRAN Symbolic Names	Mathematical Symbology	Definition			
XIP	x				
YIP	Y				
ZIP	z	ID 5000 11 /			
XIP1	$\dot{\mathbf{x}}$	IP 7090 coordinate system (See Appendix B)			
YIP1	Ý				
ZIP1	ż				
XI	ξ)				
ETA	η				
ZETA	ζ	B-GE coordinate system (i.e., GE-			
xII	ζ ξ	quasi-inertial) (See Appendix B)			
ETA1	ή				
ZETA1	ζ				
YII	$\dot{\mathtt{y}}_{\mathrm{i}}$				
XX	x				
YY	у	Intermediate rotational coordinate			
$\mathbf{Z}\mathbf{Z}$	z	system (See NASA Working Paper 146)			
XX1	ż				
YY1	ý				
ZZ1	ż				

APPENDIX C

COORDINATE CONVERSION SYSTEMS

This appendix deals with the following coordinate systems involved in the Postflight Reporter Program:

True inertial coordinates $\underline{\overline{X}}$, $\underline{\overline{Y}}$, $\underline{\overline{Z}}$, $\underline{\dot{\overline{X}}}$, $\underline{\dot{\overline{Y}}}$, $\dot{\overline{Z}}$

GE quasi-inertial coordinates ξ , η , ζ , $\dot{\xi}$, $\dot{\eta}$, $\dot{\zeta}$,

IP 7090 quasi-inertial coordinates X, Y, Z, X, Y, Z

Pad rectangular coordinates u, v, w, u, v, w

In addition, the appendix includes definitions of Miscellaneous Transformations.

TRUE INERTIAL COORDINATE SYSTEM $(\overline{X}, \overline{Y}, \overline{Z}, \dot{\overline{X}}, \dot{\overline{Y}}, \dot{\overline{Z}})$

This fundamental system is a right-handed rectangular system centered at the earth's center, $\overline{\underline{X}}$ pointing to the first point of Aries (T) or vernal equinox, $\overline{\underline{Z}}$ lying along the earth's polar axis, and $\overline{\underline{Y}}$ normal to the meridian plane. In general, data in other systems are transformed to these coordinates prior to processing. In the Postflight Reporter program, IPCNV converts IP 7090 data to true inertial reference and GECNV converts B-GE data to true inertial reference. However, MERCNV converts true inertial coordinates to pad rectangular coordinates.

GE QUASI-INERTIAL COORDINATE SYSTEM $(\xi,\eta,\zeta,\dot{\xi},\dot{\eta},\dot{\zeta})$

At two-inch lift-off, this system has the following configuration: ξ and η are axes lying in the earth's equatorial plane, with ξ lying in the meridian plane of the GE central radar, ξ is the normal lying along the earth's polar axis. The origin is at the earth's center and this system is a rectangular right-handed set of axes.

If $\delta \phi_{\rm R}$ is the angle between the meridian plane through the first point of Aries (7) and the ξ , ζ plane at two-inch lift-off, then with terrestrial polar rotation rate of $\omega_{\rm e}$ radians/second after t seconds, this angle is changed by

 $\omega_{\rm e}$ t to the value $\delta \phi_{\rm R}^{+} \omega_{\rm e}$ t. Therefore, if $\nu_{\rm a}$ is the Greenwich mean hour angle at two-inch lift-off and if $\lambda_{\rm 0}$ represents the longitude of the GE central radar, then $\delta \phi_{\rm R} = \nu_{\rm a} + \lambda_{\rm 0}$ (actually, $\lambda_{\rm 0}$ is negative for Western longitudes).

Since the relationship between the GE and true inertial frames is merely a rotation about the common \overline{Z} or ζ axis of angle $\delta \phi_R + \omega_e t$, the transformation may be written.

$$\begin{pmatrix} \overline{\underline{X}} \\ \overline{\underline{Y}} \\ \overline{\underline{Z}} \end{pmatrix} = \begin{pmatrix} \cos (\delta \phi_{R} + \omega_{e} t) & -\sin (\delta \phi_{R} + \omega_{e} t) & 0 \\ \sin (\delta \phi_{R} + \omega_{e} t) & \cos (\delta \phi_{R} + \omega_{e} t) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \xi \\ \eta \\ \zeta \end{pmatrix}$$

Thus, $\vec{r}_{true} = M\vec{r}_{GE}$ where M is the above matrix. The velocity relationship is obtained by differentiation, $\vec{V}_{true} = M \vec{V}_{GE}$, or explicitly

$$\begin{pmatrix} \frac{\dot{\overline{X}}}{\overline{X}} \\ \frac{\dot{\overline{Y}}}{\overline{Z}} \end{pmatrix} = \begin{pmatrix} \cos (\delta \phi_{R}^{+} \omega_{e}^{t}) & -\sin (\delta \phi_{R}^{+} \omega_{e}^{t}) & 0 \\ \sin (\delta \phi_{R}^{+} \omega_{e}^{t}) & \cos (\delta \phi_{R}^{+} \omega_{e}^{t}) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \dot{\xi} \\ \dot{\eta} \\ \dot{\zeta} \end{pmatrix}$$

IP 7090 QUASI-INERTIAL COORDINATE SYSTEM (X, Y, Z, X, Y, Z)

At two-inch lift-off, this system has the following configuration: X lies in the earth's meridian plane through Greenwich and in the equatorial plane. Z lies in the polar axis. Y lies in the equatorial plane normal to the X, Z plane and such that X, Y, Z forms a right-handed system.

If ν_a is the Greenwhich hour angle at two-inch lift-off, then with terrestrial angular rotation rate about the polar axis of ω_e radians/second, after t seconds this angle is changed by $\omega_e t$ to the value ν_a^+ $\omega_e t$.

The relation between IP and true inertial coordinates is analagous to the GE and true inertial relationship. The conversion between systems is given by the following transformation:

$$\begin{pmatrix} \overline{X} \\ \overline{Y} \\ \overline{Z} \end{pmatrix} = \begin{pmatrix} \cos \left(\nu_{a} + \omega_{e} t\right) & -\sin \left(\nu_{a} + \omega_{e} t\right) & 0 \\ \sin \left(\nu_{a} + \omega_{e} t\right) & \cos \left(\nu_{a} + \omega_{e} t\right) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

$$\begin{pmatrix} \overline{X} \\ \overline{Y} \\ \vdots \end{pmatrix} = \begin{pmatrix} \cos \left(\nu_{a} + \omega_{e} t\right) & -\sin \left(\nu_{a} + \omega_{e} t\right) & 0 \\ \sin \left(\nu_{a} + \omega_{e} t\right) & \cos \left(\nu_{a} + \omega_{e} t\right) & 0 \\ \vdots \end{pmatrix} \begin{pmatrix} \dot{X} \\ \dot{Y} \\ \vdots \end{pmatrix}$$

PAD RECTANGULAR COORDINATE SYSTEM (u, v, w, u, v, w)

This is a rectangular coordinate system with origin at the launch pad. The u axis points downrange, the w axis is normal to the tangent plane at the pad and directed away from the surface, and the v axis is normal to the u, w plane and directed eastward.

Since the system is hinged at the surface, a combination of rotations and translations is necessary to convert from true inertial to the pad rectangular system. If $\delta \theta_p$ is the hour angle (at two-inch lift-off) of the launch pad, then $\delta \theta_p = \nu_a + \lambda_{PAD}$ ($\lambda_{PAD} < 0$ for western longitudes). Then, as in previous cases, after t seconds, $\vec{r}_{rot} = M\vec{r}_{inertial}$ is used to pass from true inertial to rotational. To translate from geocenter to pad, $\vec{r}_{rot} = M\vec{r}_{inertial} + \vec{r}_{0}$.

In the tangent plane, rotation puts \vec{r}_{rot} relative to the equatorial plane—normal $\vec{r}_{rot}' = \vec{Nr}_{rot}$ through L_{ρ} (geocentric latitude of the pad). Finally, to rotate from east-north in tangent plane to downrange vs. crossrange through the angle θ_0 (planned azimuth) $\vec{r}_{pad} = \vec{Lr}_{rot}'$. The matrices for the transformations are:

$$\mathbf{M} = \begin{pmatrix} \cos \left(\delta \theta_{\mathbf{p}} + \omega_{\mathbf{e}} \mathbf{t}\right) & \sin \left(\delta \theta_{\mathbf{p}} + \omega_{\mathbf{e}} \mathbf{t}\right) & 0 \\ -\sin \left(\delta \theta_{\mathbf{p}} + \omega_{\mathbf{e}} \mathbf{t}\right) & \cos \left(\delta \theta_{\mathbf{p}} + \omega_{\mathbf{e}} \mathbf{t}\right) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\vec{\mathbf{r}}_{0'} = - \begin{pmatrix} \mathbf{r}_{0'} & \cos \mathbf{L}_{\rho} \\ \mathbf{0} \\ \mathbf{r}_{0'} & \sin \mathbf{L}_{\rho} \end{pmatrix} \quad \text{where } \mathbf{r}_{0'} = \mathbf{\bar{R}}_{\mathbf{p}} + \mathbf{h}_{\mathbf{0}}$$

for geocentric pad latitude L_{ρ} (= tan⁻¹ $\left[\left(\frac{b}{a_c} \right)^2 \tan L_{p'} \right]$, $L_{p'}$ = geodetic).

$$\vec{R}_{p} = \sqrt{\frac{a_{c}}{\cos^{2} L_{\rho} + \left(\frac{a_{c}}{b_{c}}\right)^{2} \sin^{2} L_{\rho}}}$$

 h_0 = height of the pad above mean sea level

$$\mathbf{N} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \mathbf{L}_{p'} & -\sin \mathbf{L}_{p'} \\ 0 & \sin \mathbf{L}_{p'} & \cos \mathbf{L}_{p'} \end{pmatrix}$$

$$L = \begin{pmatrix} \sin \theta_0 & \cos \theta_0 & 0 \\ -\cos \theta_0 & \sin \theta_0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

To obtain the velocity relation, the translation and rotation matrices are compounded and differentiated to find,

$$\begin{pmatrix} \dot{\mathbf{u}} \\ \dot{\mathbf{v}} \end{pmatrix} = \begin{pmatrix} \sin\theta_0 & \cos\theta_0 & 0 \\ -\cos\theta_0 & \sin\theta_0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos L_{p'} & -\sin L_{p'} \\ 0 & \sin L_{p'} & \cos L_{p'} \end{pmatrix}$$

$$\begin{pmatrix} \cos\left(\delta\theta_p + \omega_e \mathbf{t}\right) & \sin\left(\delta\theta_p + \omega_e \mathbf{t}\right) & 0 \\ -\sin\left(\delta\theta_p + \omega_e \mathbf{t}\right) & \cos\left(\delta\theta_p + \omega_e \mathbf{t}\right) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \dot{\overline{\mathbf{X}}} + \omega_e & \overline{\mathbf{Y}} \\ \dot{\overline{\mathbf{Y}}} - \omega_e & \overline{\mathbf{X}} \\ \dot{\overline{\mathbf{Z}}} \end{pmatrix}$$

MISCELLANEOUS TRANSFORMATIONS

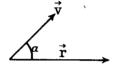
To obtain the geocentric latitude L_C from the geodetic latitude L_D with respect to a spheroid of equatorial radius, a, polar radius, b, the relation tan $L_C = \left(\frac{b}{a}\right)^2 \tan L_D$ is used. To find the geocentric distance \overline{R}_C to a point on the spheroid whose geocentric latitude is L_C , $\overline{R}_C = \sqrt{\cos^2 L_C + \left(\frac{a}{b}\right)^2 \sin^2 L_C}$ is used.

Position in geocentric spherical coordinates (r, L_C , λ) may be expressed in (geocentric) true inertial frame by the relationships $\overline{\underline{X}} = r \cos L_C \cos \lambda$, $\overline{\underline{Y}} = r \cos L_C \sin \lambda$, $\overline{\underline{Z}} = r \sin L_C$.

Derivations for data used in the Postflight Reporter processing programs and their subroutines are shown below.

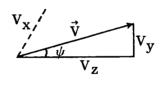
Flight path angle, y

$$\vec{r} \cdot \vec{v} = rv \cos \alpha$$
 but $\alpha = 90^{\circ} - \gamma$
= $rv \sin \gamma$

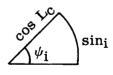


Heading angle, ψ

Project \overline{V} onto YZ plane, $\tan \psi = \frac{V_y}{V_z}$



In a spherical triangle, $\sin \psi_i = \frac{\sin_i}{\cos L_C}$



Mach number, M

This is defined as the relative speed - local speed of sound ratio.

M = V_e / $C_{S'}$, where $|\vec{v}_e| = |\vec{v}_i - \vec{\omega}_e|$ x \vec{r} | is the speed with respect to rotating atmosphere.

Reynolds number (per foot)

This is defined as the ratio of relative speed to local kinematic viscosity.

$$R_N = V_e / \nu$$
.

Dynamic pressure

Dynamic pressure is the ratio of drag to drag coefficient. Therefore, $q_D = 1/2 \rho \ V_e 2$, where ρ is the local atmospheric density.

Stagnation heating rate

This is a function of altitude - for free stream q_1 = K_ρ u^3 for $q = K_1 \sqrt{\rho} \ u^{3.15}$

where K_1 are given and u is free stream speed.

Downrange distance (RFLP)

The formula for downrange distance is the spherical trigonometric relation (from the spherical triangle), $\theta = \cos^{-1} \left[\sin L_D \sin L_\rho + \cos L_D \cos L_\rho \cos (\lambda - \lambda_p) \right] \left[\frac{\overline{R}_p + \overline{R}}{2} \right]$.

APPENDIX D REPORT DATA FORMATS

This appendix presents the data formats for each of the postflight reports. In accordance with specifications of NASA, the formats and heading comprise the key for reporting of postflight data. Each report presents, in a specific time interval, selected data about each applicable phase of the mission, as follows:

Phase	Quick Look	Three-Day Report	Condensed Report		
Launch	4 seconds	0.5 seconds	5 seconds		
Abort	4 seconds	1.0 seconds	10 seconds		
Orbit	2 minutes	24.0 seconds	1 minute		
Re-entry	12 seconds	6.0 seconds	0.5 minute		

At the beginning of each phase, a heading is printed, giving the name of the phase, the GMT of lift-off, and the longitude of Aries at lift-off, in degrees. After each heading there is a paragraph containing postflight information about the phase under investigation. The data formats for each report are shown below. (The mathematical symbology, which is defined in Appendix A, represents digital data from the Postflight Reporter Program.)

QUICK LOOK DATA FORMATS

Launch

Line 1	t, sec	u	v	w	ù	v	ŵ
2		X	Y	${f z}$	x	Ÿ	Ż
3		$\mathbf{v_i}$	$\gamma_{\mathbf{i}}$	$\psi_{\dot{1}}$	^{L}C	λ	h _e
4	t, hrs	v_e	γ _e	$^{\psi}\mathrm{_{e}}$	^{L}D	r	(r - R)
5	t, min	d	y	v/v_R	$^{ m A}_{ m R}$	S	
6	t, sec	M	^{q}D	$\mathbf{R}_{\mathbf{N}}$	$^{\mathbf{q}}\mathbf{s}$		

Ab	ort
210	O I 0

Line 1	t, sec	u	v	w	ů	Ÿ	ŵ
2		X	Y	${f z}$	x	Ý	Ż
3		v_{i}	$\gamma_{\mathbf{i}}$	$\psi_{\mathbf{i}}^{}$	$^{\mathrm{L}}\mathrm{_{C}}$	λ	h _e
4	t, hrs	v_e	γ _e	$\psi_{\mathbf{e}}$	$\mathbf{r}^{\mathbf{D}}$	r	(r - R)
5	t, min	d	y		$^{\mathrm{A}}_{\mathrm{R}}$	S	
6	t, sec	M	$^{ m q}_{ m D}$	R_{N}	$^{ m q}_{ m S}$		
Orbit							
	t, min	X	Y	${f z}$	x	Ý	Ż
2		$\mathbf{v_i}$	$\gamma_{\mathbf{i}}$	$\psi_{f i}$	L _C	λ	h _e
3		v_{e}	γ _e	${^\psi}_{ m e}$	$^{\mathrm{L}}_{\mathrm{D}}$	r	(r - R)
4		Blank	Ü	J	_		
5		$\mathbf{t_p}$	T	e	i	ω	ώ
6		a – \overline{R}	$^{ heta}1$	E	M	Ω	$\dot{\Omega}$
Re-ent:	rv		-				
				_	•_	<u>.</u>	÷
Line 1	t, min	X	Y	${f Z}$	X	Ÿ	Ż
2		$\mathbf{v_i}$	$\gamma_{\mathbf{i}}$	$\psi_{\mathbf{i}}$	^{L}C	γ	^h e
3		v_{e}	$^{\gamma}e$	${\psi}_{\mathbf{e}}$	$^{\mathrm{L}}\mathrm{_{D}}$	r	(r - \overline{R})

THREE-DAY REPORT DATA FORMATS

M

 $\mathbf{q}_{\mathbf{D}}$

Launch

4

Lines 1 through 6 are identical to the Launch format for the Quick Look.

 $\mathbf{R}_{\mathbf{N}}$

 s_R

 $^{\mathbf{q}}_{\mathbf{S}}$

Line 7
$$\phi_{\min}$$
 λ_{\min} ϕ_{\max} λ_{\max} i Area

8 GO-NOGO
$$\Delta t_{R}$$
 GMTRC

9
$$(V/V_R-V/V_R^{nom})GE$$
 $(\gamma-\gamma_{nom})GE$ $(V/V_R-V/V_R^{nom})IP$ $(\gamma-\gamma_{nom})IP$

Abort

Lines 1 through 6 are identical to the Abort format for the Quick Look.

8 Area
$$\phi_{\text{IP}}$$
 λ_{IP}

Orbit

Lines 1 through 6 are identical to the Orbit format for the Quick Look.

Line 7 Orbit number Orbit capability
$$h_a$$
 Area ϕ_{ID} λ_{ID}

8
$$(GMTRC)_1 (ECTRC)_1 (GMTRC)_2 (ECTRC)_2 (GMTRC)_3 (ECTRC)_3$$

9 GMTRS ICTRC GTRS GMTLC
$$\lambda_{p}$$

Re-entry

Lines 1 through 4 are identical to the Re-entry format for the Quick Look.

Line 5 Area
$$\phi_{ ext{IP}}$$
 $\lambda_{ ext{IP}}$ EGT GTL GMTLC

CONDENSED REPORT DATA FORMATS

Launch

Line 1 t, sec
$$V_i$$
 γ_i $(r-\overline{R})$ V/VR area
$${}^2 \qquad \qquad L_D \quad \lambda \qquad M \qquad \qquad q_D$$

MC 63-4

Abort

Line 1 t, sec
$$V_i$$
 γ_i $(r - \overline{R})$ L_D λ

2 M
$$Q_{\mathrm{D}}^{\phantom{\mathrm{D}}}$$
 $\phi_{\mathrm{IP}}^{\phantom{\mathrm{D}}}$ $\lambda_{\mathrm{IP}}^{\phantom{\mathrm{D}}}$

<u>Orbit</u>

Line 1 t, min
$$V_i$$
 γ_i $(r - \overline{R})$ h_a area (ECTRC)₂

Re-entry

Line 1 t, min
$$V_i$$
 γ_i $(r - \overline{R})$ L_D λ

2
$$M = Q_D = \phi_{IP} = \lambda_{IP}$$